Spectrum +

MANUAL DE INSTRUCCIONES

(Textos en Castellano)

SINCLAIR ZX SPECTRUM
****

Contenido de este volumen

-- MANUAL DE INTRODUCCION

-- MANUAL DE PROGRAMACION EN BASIC

CONTENIDO MANUAL DE INTRODUCCION

CONTENIDO DEL MANUAL DE PROGRAMACION EN BASIC.-

P R E F A C I O

Con la aparición en el mercado español del nuevo Spectrum +, Sinclair ha optado por una posición intermedia, a medio camino entre un modelo completamente nuevo y que, por tanto,hubiera supuesto un largo proceso de consolidación antes de su aceptación en el mercado y el continuismo puro y simple, totalmente suicida en un momento en que, como el actual, raro es el mes en que no aparece un nuevo producto informático cada vez más barato y mejor que el anterior.

El Spectrum +, en nuestro pais, ha sido anunciado, siguiendo la tónica de otros fabricantes, como poseedor de 64 K. de memoria , cuando lo cierto es que la capacidad de memoria RAM del mismo (la única disponible por el usuario) es de 48 K. Con esta política se ha tratado de presentar el nuevo Spectrum como algo distinto del antiguo cuando lo cierto es que se trata del mismo ordenador (misma placa interior, mismas especifícaciones), siendo ambos, por tanto, totalmente compatibles con el mismo software.

La única e importante diferencia entre ambos está en el teclado y en la distinta distribución de las teclas, siendo el del Spectrum + bastante más racional que el del Spectrum normal.

Cuando decimos que el Spectrum + es idéntico al Spectrum de 48 K. con el único aditamento del teclado, no hacemos ninguna comparación peyorativa. Estamos convencidos de que el Spectrum se adelantó a su época y aúnque últimamente han salido productos de alta calidad procedentes de otros fabricantes, hoy por hoy el Spectrum y su sucesor, el Spectrum + siguen siendo imbatibles en el mercado del ordenador barato, sencillo y potente a la vez, con una impresionante cantidad de software disponible, con varias revistas a ellos dedicadas exclusivamente y con un soporte técnico de firmas independientes que fabrican accesorios y periféricos de alta calidad y precio ajustado, que para sí desearían muchos otros ordenadores que se autotitulan "profesionales".

Este manual, previsto originalmente para el Spectrum de 48 K., se ha suplementado con aquellos conceptos que podían quedar más oscuros, en especial la forma de conexión, la forma de guardar programas en casette y la de cargar programas en la memoria. También se ha dedicado un capítulo al estudio del teclado que, como ya hemos dicho, es el único elemento del Spectrum + que varía respecto al normal. Se acompaña también el manual en inglés, válido para copiar programas.

EL TECLADO : PANEL DE CONTROL DE SU SPECTRUM +

Como ya sabrá, el ZX Spectrum + emplea el lenguaje BASIC que es uno de los 'dialectos' más potentes empleado por la inmensa mayoría de los ordenadores personales. Para hacer que el ordenador ejecute sus instrucciones debe Vd. redactar un programa en BASIC empleando, para ello, el teclado que sirve para que el ordenador 'tenga conocimiento' de las órdenes. Básicamente, el usuario dialoga con el ordenador a través del teclado y éste contesta de alguna manera mediante la pantalla del televisor.

Todos los ordenadores que emplean BASIC difieren entre sí por la utilización de unos 'subdialectos' del BASIC qué, aúnque iguales en lo fundamental, tienen unos matices diferentes que los hacen incompatibles entre sí. Es decir, un programa para un ordenador no puede ejecutarse en otro ordenador sin efectuar antes unos pequeños cambios en el programa.

Una de las características más importantes del Basic del Spectrum + y que facilita la tarea de redactar un programa, es el sistema de introducción de instrucciones o 'keywords' mediante una tecla única.

TECLAS Y PALABRAS CLAVE.-

Las palabras clave (o keywords), son palabras con un significado especialmente definido en BASIC, cada una de ellas es una instrucción específica para el ordenador, como, por ejemplo, PRINT o INPUT. En la mayoría de los ordenadores es necesario introducir las palabras clave letra por letra, como si fuera una máquina de escribir, y es necesario verificar que no se cometan errores.Pero en el Spectrum basta con oprimir la tecla indicada para obtener una palabra clave completa.

El BASIC de Sinclair tiene más de 8Ø palabras clave, las cuales son introducidas mediante un total de 36 teclas (26 teclas-letra y 10 teclas número).Muchas de las teclas tienen la capacidad de producir varias palabras clave que son reconocidas por el ordenador. La mayoría de las teclas le suministran una palabra clave determinada o una letra, número, signo o aún una forma (cárácteres gráficos), todos los cuales pueden ser utilizados en sus programas.

SELECCIONANDO PALABRAS CLAVE Y SIMBOLOS.-

En el teclado del Spectrum + se encuentran dos teclas que utilizará con mucha frecuencia, éstas son :EXTEND MODE y SYMBOL SHIFT.El propósito de estas teclas es ayudar a seleccionar las palabras clave y signos que usted desea hacer aparecer en la pantalla, operándolas en combinación con las demás teclas. En primer lugar, le sugerimos que estudie la distribución del teclado. Las siguientes dos páginas le demostrarán exactamente como seleccionar cualquiera de las instrucciones que aparecen indicadas en el teclado del ordenador. Después de haber aprendido todo esto, usted estará ya listo para escribir sus propios programas.

COMO OPERAR LAS TECLAS.

Es posible obtener hasta seis palabras claves, letras, números o signos de la mayoría de las teclas de su ZX Spectrum +. Sin embargo, seleccionar un carácter o palabra clave en el teclado no es una operación difícil, una vez que usted se familiarice con las características especiales de su ordenador. El resultado obtenido al oprimir una tecla dependerá del

"modo" en el cual el ordenador se encuentra en ese momento.Cada modo determina que la tecla produzca una información diferente como, por ejemplo una letra, una palabra clave o un carácter gráfico. La ventaja de este sistema es que el Spectrum + le ayudará a seleccionar el modo apropiado, para introducir en el orden correcto, instrucciones e información en el ordenador.

MODO DE PALABRA CLAVE.-

Conecte y reajuste su Spectrum + para que aparezca el mensaje inicial. Ahora pulse la tecla ENTER. El ordenador hará aparecer una letra K intermitente al pie de la pantalla.Este cuadro luminoso es el cursor.El cursor le señala el punto donde algo aparecerá en la pantalla, y la letra K le advierte que el ordenador se encuentra en el modo de palabra clave.Oprima cualquier tecla-letra y la palabra clave situada en la parte de arriba de la tecla, aparecerá en la pantalla. Por ejemplo, pulse la letra Q, y la palabra clave PLOT aparecera en la pantalla.Pulse la tecla DELETE para borrar esa palabra clave y pruebe las demás teclas. Las teclas-número le darán números, pero tan pronto como usted pulse una tecla-letra,aparecerá la palabra clave de la parte de arriba de la tecla.

Utilize DELETE otra vez para volver a hacer aparecer el cursor K. A continuación pulse cualquiera de las teclas SYMBOL SHIFT, manténgala pulsada y apriete cualquier otra tecla-letra.Ahora aparecerá la palabra clave o signo ubicado justo encima de la letra, en la sección elevada de la tecla. En el caso de las teclas-número aparecerá el signo a la derecha en la sección elevada.

MODO DE LETRAS Y MAYUSCULAS.-

Después de haber producido una palabra clave o signo operando en el modo Palabra Clave, el ordenador automáticamente cambia el modo del cursor al Modo L. El Modo L es también conocido como el Modo Letra (letter mode). Cuando en el Modo L, usted pulse una tecla-letra, inmediatamente se visualiza dicha letra en minúscula. Igualmente, si pulsa una tecla-número aparecerá el número correspondiente. Si desea obtener letras mayúsculas , bastara con pulsar la tecla CAPS SHIFT (cambio de mayúsculas) seguida de la tecla con la letra en cuestión.

Si desea escribir solamente con letras mayúsculas, oprima la tecla CAPS LOCK. El cursor cambia a la letra C. Su Spectrum + se encuentra ahora en el Modo de Mayúsculas (capitals mode): cada vez que usted pulse una tecla-letra el ordenador imprimirá una letra mayúscula en la pantalla. El procedimiento para retornar al Modo Letra (L) es simplemente pulsar nuevamente la tecla CAPS LOCK.

EXTENDED MODE.-

El siguiente modo es denominado Extended Mode y es introducido presionando la tecla EXTENDED MODE. Como resultado el cursor cambia a la letra E.Después de haber cambiado a este modo pulse cualquier tecla-letra y podrá apreciar que el ordenador presenta en la pantalla la palabra clave superior, del par de palabras clave situadas encima de la sección elevada de la tecla. Por ejemplo: ponga el ordenador en el Extended Mode y luego pulse la tecla B, el resultado en la pantalla será la palabra clave BIN. Para obtener la palabra clave inferior, o el signo encima de la sección elevada, el procedimiento es:(1) pulsar cualquiera de las teclas SYMBOL SHIFT, (2) mantener la tecla en la posición apretada, (3) y pulsar la tecla-letra correspondiente. Por ejemplo, en el caso de la tecla-letra B, siguiendo este procedimiento se obtiene la palabra clave BRIGHT. Después de haber sido pulsada una tecla (La tecla EXTEND MODE)mientras el ordenador se encuentra en Extend Mode, éste automáticamente retornará al Modo Letra o al Modo Mayúsculas.

MODO GRAFICO.-

El quinto modo es el Modo Gráfico (Graphics Mode), y es introducido mediante la tecla GRAPH. A continuación el cursor cambia a la letra G. Pulse cualquiera de las teclas-número, del 1 al 8, y podrá ver como el carácter gráfico correspondiente aparece en la pantalla. A continuación pulse la tecla CAPS SHIFT y cualquier tecla-número, del 1 al 8.Los símbolos gráficos aparecen nuevamente, pero esta vez los colores blanco y negro han sido invertidos. Para abandonar el Modo Gráfico se pulsa otra vez la tecla GRAPH.

EDITANDO PROGRAMAS.-

Cuando usted escriba sus propios programas para su Spectrum + , deseará corregir los errores, ya sea en comandos o en líneas del programa, o efectuar alteraciones. Usted puede hacerlo fácilmente editando el programa.

COMO CORREGIR UN ERROR.-

Cuando se introduce en el ordenador una línea en BASIC con errores o un comando equivocado,el Spectrum mostrará en la pantalla un ? destellante delante del error. Para corregir el error, oprima la tecla de control de cursor de la derecha o izquierda para mover el cursor a la derecha del error. A continuación borre el error pulsando la tecla DELETE , o agregue los elementos necesarios. Finalmente oprima la letra ENTER.

Por ejemplo, suponga que desea que su ordenador multiplique 7 por 8 pero se olvida de pulsar la tecla SYMBOL SHIFT para obtener el símbolo *. Como consecuencia usted habría introducido.

PRINT 7b8

Su Spectrum no puede obedecer esta instrucción, por lo tanto cuando usted pulse la tecla ENTER, el ordenador mostrará un signo de interrogación destellante delante de la b, el lugar donde se encuentra el error. Todo lo que usted tiene que hacer es: mover el cursor justo a la derecha del error,oprimir la tecla DELETE para borrar la b; pulsar la tecla SYMBOL SHIFT y la tecla B para obtener * ; y pulsar la tecla ENTER para que el ordenador obedezca el comando correcto. No es necesario mover el cursor hasta el final de la línea.

COMO EDITAR UNA LINEA DE PROGRAMA.-

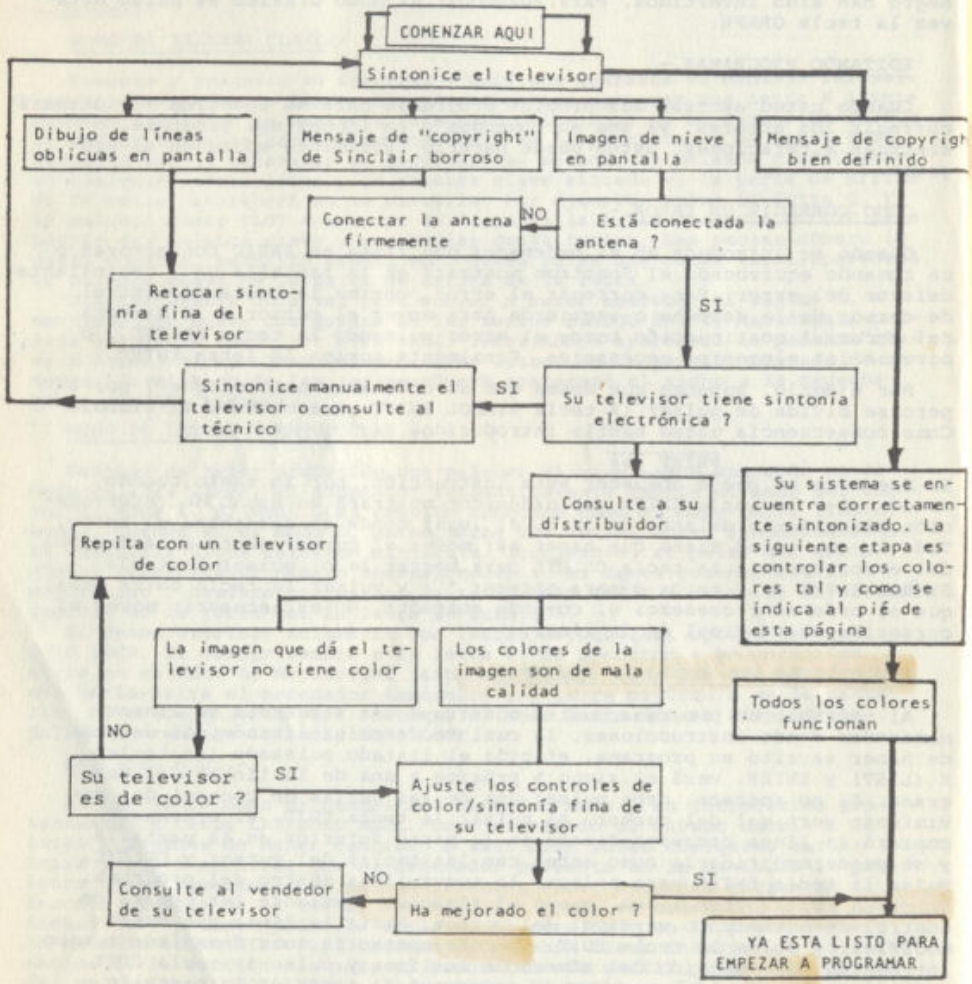
Al escribir un programa, usted construye una secuencia de líneas numeradas dando instrucciones, la cual se denomina listado. Si después de haber escrito su programa, efectúa el listado pulsando las teclas K (LIST) y ENTER, verá el signo > próximo a una de la líneas del programa. Si no aparece, debe pulsar una de las teclas de control del movimiento vertical del cursor. Al pulsar la tecla EDIT, el ordenador copiará la línea correspondiente en la parte inferior de la pantalla y se puede modificarla como antes con las teclas del cursor y DELETE. Pulse la tecla ENTER para colocar la nueva línea dentro del programa. Si desea editar otra línea, mueva el símbolo > mediante las teclas de control de movimiento vertical del cursor, hasta la línea que desea modificar y pulse la tecla EDIT. Si esta operación toma demasiado tiempo, introduzca LIST seguido del número de la línea y pulse la tecla EDIT. En cada caso , la línea requerida aparecerá al fondo de la pantalla y se le puede cambiar.

Para borrar una línea completa del programa, introduzca el número de la línea y pulse la tecla ENTER.Si introduce un programa que contiene un error ,un informe de error aparecerá en la pantalla.

Ver el apéndice B para más detalles sobre estos informes.

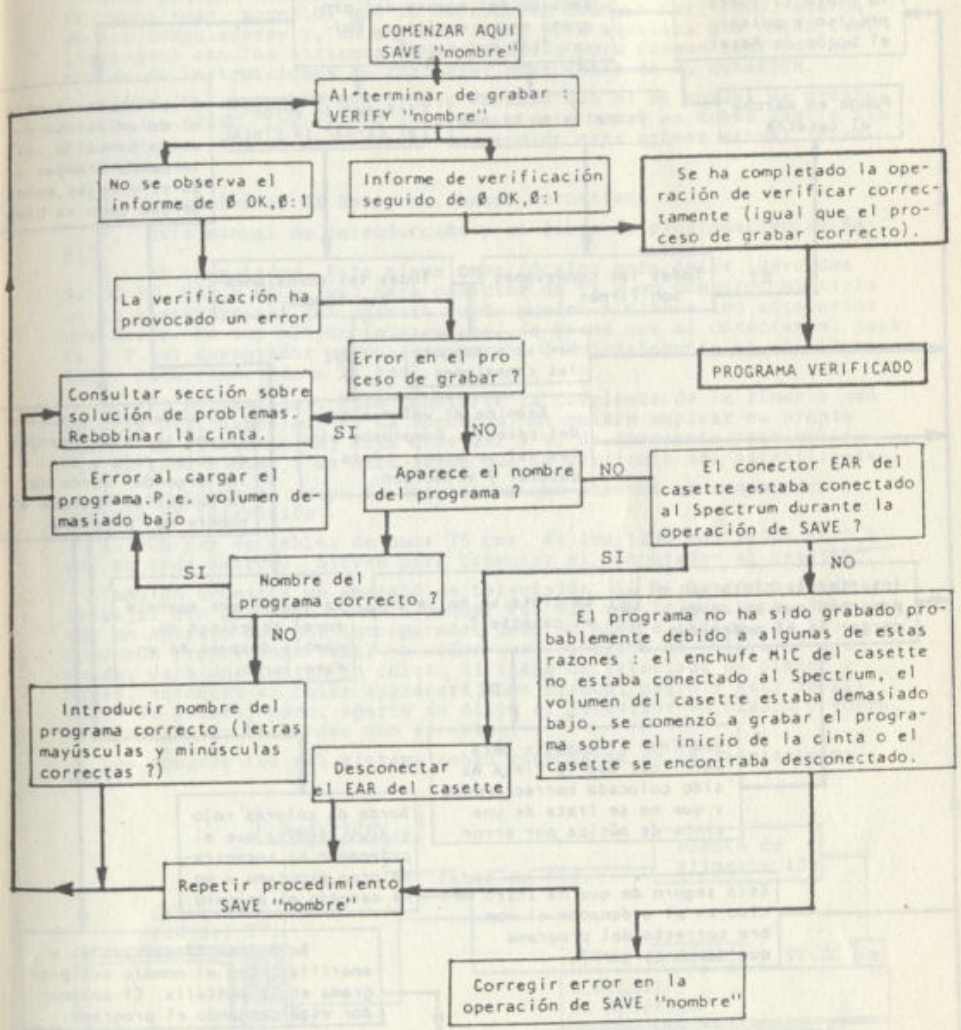
--- FORMA DE CONECTAR EL ZX SPECTRUM + ---

Lea detenidamente la Página I de la Introducción, donde está detalladamente explicada la forma de conectar entre sí todos los elementos que forman el sistema. A continuación hay un diagrama detallado de los pasos a efectuar para solucionar los problemas preliminares de puesta en marcha.

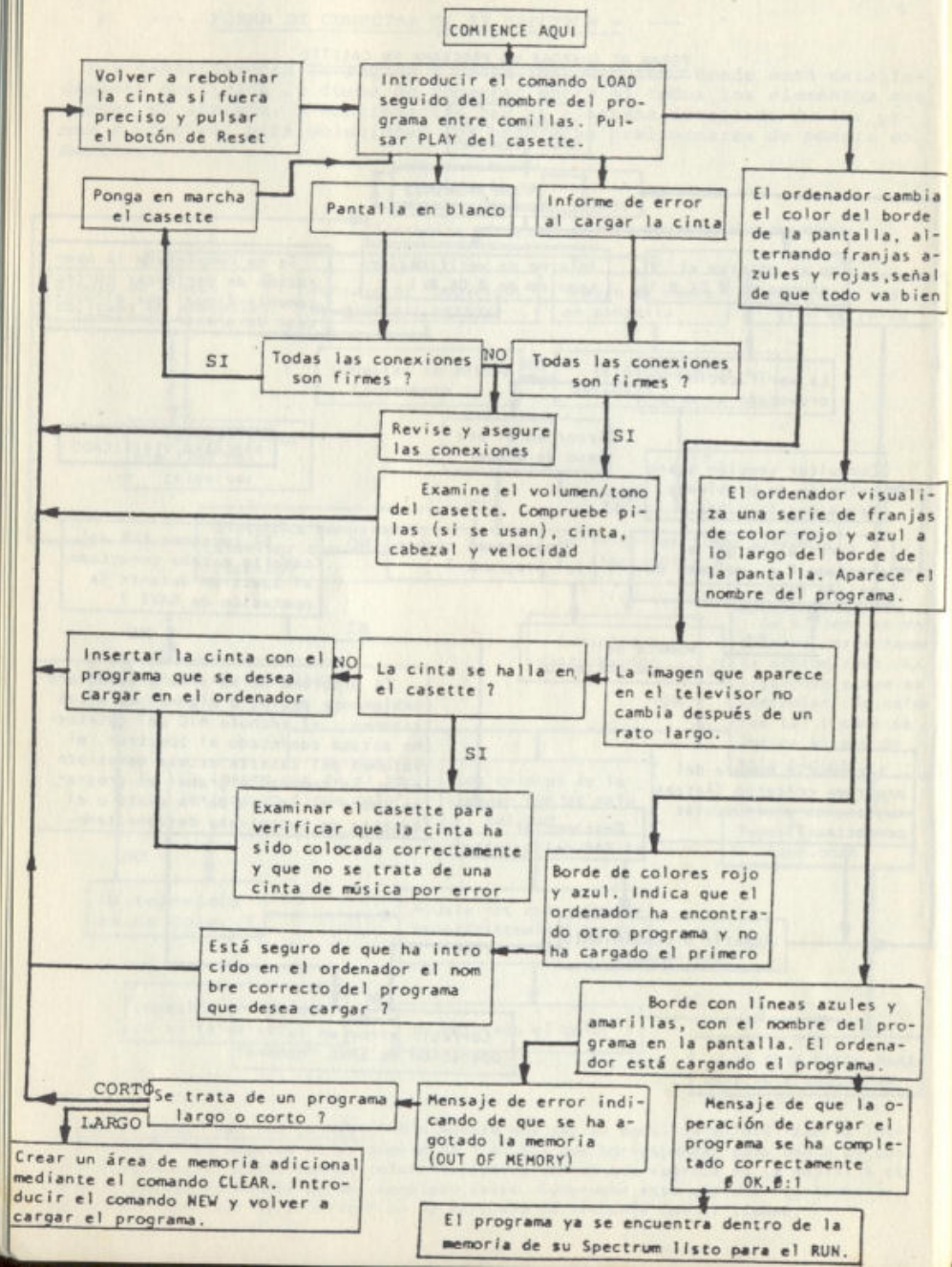


Forma de comprobar los colores.- Una manera muy rápida consiste en utilizar la instrucción BORDER seguida de un número de 1 al 6 (que corresponden cada uno a un color distinto).Si coinciden el color respectivo con el que aparece en la pantalla es que la puesta en marcha es un completo éxito. Compruebe este sistema regularmente para verificar que su televisor no ha marchado de sintonía con el tiempo.

FORMA DE GUARDAR UN PROGRAMA EN CASETTE



FORMA DE CARGAR UN PROGRAMA DESDE EL CASETTE



1°.- EL COMPUTADOR Y SU PUESTA EN FUNCIONAMIENTO.-

Este pequeño manual ha sido escrito pensando en dos tipos de gente. Ante todo, aquellos que lo desconocen todo, o casi todo, acerca de los computadores y, en segundo lugar para aquellos que están familiarizados con los sistemas de computación pero desean leer los manuales de instrucciones de cualquier cosa antes de su conexión.

Existe un segundo y más grueso volumen que es el manual de programación en BASIC. Los recién llegados a este campo no deben leerlo sin antes estar seguros de que han comprendido este primer manual introductorio.

Al desembalar el ZX Spectrum, habrá encontrado .

1.- Este manual de introducción y el libro de programación en BASIC.

2.- El computador. Este tiene tres zócalos para jacks (marcados 9V DC IN, EAR y MIC), uno para conector de TV y un conector múltiple en la parte trasera del aparato donde pueden alojarse los accesorios que desee. No hay ningún interruptor, de forma que al conectar el jack de 9 V. el computador se pone en marcha (evidentemente si el alimentador está conectado a la red).

3.- El alimentador. Este convierte la corriente de la línea o red en otra utilizable por el ZX Spectrum. Si quiere emplear su propia fuente de alimentación, ésta debe dar 9 V. de corriente continua y 1,4 A. como mínimo (no hace falta que la corriente sea estabilizada).

4.- Un cable de cerca de dos metros que conecta el computador al aparato de televisión .

5.- Un par de cables de unos 75 cms. de longitud con jacks de 3,5 mm. en cada extremo. Sirven para conectar el computador al casette.

También necesita un aparato de televisión. El ZX Spectrum puede trabajar sin él, pero se va a quedar Vd. sin ver lo que hace. Debe ser un aparato con UHF incorporado. Como su nombre indica, el ZX Spectrum proporciona señal en color que, si tiene un televisor adecuado, dará una imagen en color. Si tiene un televisor en blanco y negro, entonces el color aparecerá como blanco, negro y seis tonalidades de grises, pero, aparte de ello, el televisor en blanco y negro, funcionará tan bien como uno en color.

Los componentes del sistema deben conectarse de la siguiente manera :

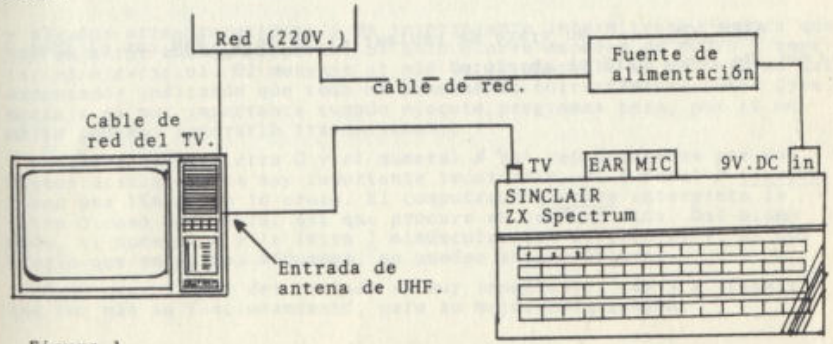


Figura 1

Si su televisión tiene dos entradas de antena señaladas con UHF y VHF, debe emplear la primera. S emplea un televisor en blanco y negro de modelo antiguo debe emplear un transformador de impedancias de 75 a 300 ohms, de venta en cualquier tienda de electrodomésticos. En caso de un modelo nuevo o de una tele en color la conexión es directa.

Conectar el computador y la Televisión. Ahora sólo le falta sintonizar ésta última. El ZX Spectrum opera en el canal 36 de UHF y, la primera vez que funciona dá la iguiente imagen :

© 1982 Sinclair Research Ltd

Cuando use el computador, probablemente tendrá que bajar al mínimo el volumen de la televisión.

Si su televisor es de sintonía variable, debe ajustarlo hasta que obtenga la inscripción dada anteriormente. La mayoría de los televisores actuales tienen un pulsador individual para cada canal Escoja uno que no se use y sintonice e computador en él

El Spectrum funciona con el Sistema de 625 líneas y frecuencia de 50 ciclos y en sistema de color PAL, sistemas empleados en la mayor parte de los países occidentales excepto en Francia que usa SECAM) En Estados Unidos, Canadá y Japón, emplean un sistema de televisión diferente y requieren una versión especial del Spectrum. En España, como hemos dicho, el ZX Spectrum es totalmente compatible con el sistema empleado

Cuando Vd. desconecta el computador, toda la información que tiene almacenada se pierde. Una forma de guardarla para más adelante, es grabarla en casette. También puede, además, comprar cintas grabadas por otra gente y ejecutar sus programas. El cable doble con dos jacks en cada extremo se emplea para conectar un casette no mal al ZX Spectrum. En el capítulo 8°de este manual veremos ésto mas detalladamente.

Ahora que ha puesto en marcha el computador, deseará usarlo. El resto de este manual introductorio le explicará como hacerlo , pero dominado por la impaciencia, habrá empezado ya a apretar teclas y descubierto que éstas hacen desaparecer el mensaje inicial Esto es bueno no puede perjudicar al computador de ninguna manera (desde el teclado, claro) Atrévase, experimente. Si se encuentra en apuros, recuerde que siempre puede volver el computador a su estado inicial simplemente desconectando el jack de 9V. y conectándolo de nuevo. Este debe ser el último recurso puesto que se pierde, de esta forma, toda la información del computador.

ATENCION. No trate de emplear el ZX 16 K. RAM con el Spectrum. No funcionará en absoluto.

2°- EL TECLADO

El teclado del Spectrum es muy parecido al normal de una máquina de escribir. Las teclas de las letras y los números están en idéntica posición ; sin embargo, cada tecla puede realizar más de una función. En una máquina de escribir las letras aparecen en minúsculas y, usadas en conjunción con la tecla de mayúsculas (shift), aparecen en mayúsculas.. El teclado del Spectrum hace exactamente lo mismo.

Para ayudarle a interpretar la situación del teclado, una letra en negativo (blanco sobre negro , aparece en la pantalla indicando la posición que ocupará el próximo carácter que aparezca cuando apriete una tecla. Dicha letra en negativo es parpadeante para distinguirla de las otras de la pantalla y es llamada 'cursor'

Al conectar el Spectrum siempre aparece un mensaje de copyright(c), en la pantalla Pulsando cualquier tecla se obtiene siempre la palabra impresa bajo la letra, en la misma tecla (llamada 'palabra clave') que llamaremos 'keyword' de ahora en adelante. Esto es a causa de que el computador siempre espera que le entre un comando (u orden) que le ordene algo y todos los comandos deben empezar con un 'key word'. Al contrario que la mayoría de computadores, el Spectrum permite introducir keywords con sólo pulsar una tecla

Por ejemplo, si la tecla P es pulsada inmediatamente después de conectar el computador, el keyword PRINT aparecerá en la pantalla. El símbolo " que está marcado en dicha tecla aparecerá de la siguiente manera : apriete la tecla SYMBOL SHIFT, que está en la parte inferior derecha del teclado y, mientras la mantiene apretada, pulse la tecla P.

El cursor cambia ahora a L (en negativo) para indicar que espera que le entren una letra. Entre las letras "Hola". Si hubiera ya algún otro texto en pantalla, desconecte el computador y empiece de nuevo. Emplee la tecla CAPS SHIFT para obtener la H mayúscula. En general, todo lo que está escrito en blanco encima de la teclas, requiere el empleo de CAPS SHIFT para obtenerlo y todo lo coloreado en rojo se obtiene mediante SYMBOL SHIFT.

Un comando que empiece con PRINT ordena al computador escribir las letras encerradas entre comillas en la pantalla. Para que este comando sea ejecutado por el computador, debe emplearse la tecla ENTER. Una vez usada, la pantalla mostrará la palabra

Hola

y algunos otros caracteres.(Un interrogante intermitente, indica que hay un error en alguna parte. Si esto ocurre empezar de nuevo y repetir el ejercicio). El mensaje al pié de la pantalla es el informe del computador indicando que todo ha funcionado correctamente (OK). Este mensaje es muy importante cuando ejecute programas pero, por el momento podemos ignorarlo tranquilamente.

Observe que la letra O y el numeral 0 son representados por dos signos distintos. Es muy importante recordarlo. El numeral 0 siempre tiene una línea que lo cruza. El computador siempre interpreta la letra O como una letra, así que procure no confundirlos. Del mismo modo, el numeral 1 y la letra l minúscula son diferentes y, al contrario que en muchas máquinas, no pueden intercambiarse.

Dado que el modo del teclado es muy importante, vamos a resumir una vez más su funcionamiento, para su mejor comprensión.

El carácter parpadeante L (en negativo), recibe el nombre de 'cursor'. Siempre muestra el lugar de la pantalla donde el computador va a colocar el próximó carácter que se le entre. No siempre se trata de una L; si desconecta el computador y vuelve a empezar pulsando ENTER, el mensaje inicial (c Sinclair etc.) se convertirá en el cursor K. La letra del cursor indica el modo en que el computador interpretará la próxima cosa que le entre. Al principio de una línea, será una K parpadeante esperando el 'keyword'.(El mensaje inicial y los informes al pié de la pantalla deben interpretarse como una K). Un keyword es una palabra del 'vocabulario' del computador colocada al principio de un comando para darle a la máquina una idea general de io que el comando va a ordenarle. Dado que el computador está esperando un keyword al principio de la línea, cuando apriete, por ejemplo, la letra P, el computador decide no interpretarla como una P, sino como PRINT; y le avisa de que va a hacer esto mediante la letra K en el cursor. Cuando ya tiene el primer keyword, no espera ninguno más de forma que, todo lo que le entre a continuación será interpretado como letras. Para indicarle esto, el computador muestra una L en el cursor(de 'letra').

Estos distintos estados son, a menudo, llamados modos y hablaremos, en adelante, del modo keyword o del modo letras (K ó L)

Si desea entrar cierta cantidad de letras mayúsculas sin tener que apretar CAPS SHIFT continuamente, puede hacerlo pulsando previamente CAPS LOCK (CAPS SHIFT y tecla 2). Para indicar esta posición la L del cursor es sustituida por una C parpadeante (de CAPITALS, mayúsculas en inglés). Para volver a minúsculas y al cursor L, apretar CAPS LOCK de nuevo.

(Si pulsa CAPS LOCK durante el modo keyword, no observará ninguna diferencia inmediata, pero después de entrar el keyword el computador se pondrá en modo C (en lugar de L).

Además de keywords, letras, números y varias expresiones de programación y funciones científicas, el teclado también posee 8 caracteres gráficos. Estos se hallan en las teclas de 1 al 8 y pueden imprimirse en la pantalla de forma similar a las letras y los números. Para lograrlo, el teclado debe cambiar a modo gráficos. Esto se obtiene pulsando la tecla CAPS SHIFT junto con la 9. Observe que el cursor se convierte en una G Pulsando de nuevo la tecla 9 volverá el cursor al modo L.

Existe un último modo que puede adoptar el teclado. El modo extendido, indicado por una E en el cursor, se obtiene apretando conjuntamente CAPS SHIFT y SYMBOL SHIFT. Con este modo se obtienen la mayoría de las funciones científicas y de programación. Pulsando las dos teclas de nuevo, el teclado vuelve al modo L.

Incluso el más eficiente mecanógrafo o programador se equivoca al pulsar una tecla. No crea que la única manera de solucionar el error sea la desconexión del computador !. Aunque no sea un mal sistema si sólo está en la primera línea, sería un grave inconveniente si ya hubiera entrado cierta cantidad de información.

Afortunadamente, podemos emplear la tecla DELETE (Borrar) para reparar un error. Por ejemplo, pocos errores podemos tener con la simple instrucción :

```
PRINT "Hola"
```

o si ?.

Suponga que no ha empleado la tecla SYMBOL SHIFT para obtener las comillas. La pantalla mostrará

```
PRINT PHola"
```

El computador, al no encontrar comillas después del PRINT, se pre para para encontrar un número y, en su lugar, halla la letra P. Mostrará su confusión mediante un ? parpadeante al final de la línea.

Felizmente no tendrá que teclearlo todo de nuevo. En la fila superior de teclas existen cuatro flechas que apuntan en las cuatro direcciones y la palabra DELETE. Para operar estas teclas, debe pulsar CAPS SHIFT y la tecla deseada al mismo tiempo. Las teclas horizontales mueven el cursor hacia la derecha o hacia la izquierda y la tecla DELETE elimina el carácfer inmediatamente anterior al cursor.

Para corregir su línea equivocada, pulse ◆ (CAPS SHIFT y 5 conjuntamente) hasta que el cursor esté inmediatamente después de la P (entésta y la H)(Observe que si mantiene apretadas dichas teclas durante un lapso de tiempo determinado el cursor se mueve continuamente, emitiendo un pequeño'click' de sonido. De hecho, si mantiene apretada cualquiér tecla durante más de tres segundos se efectúa la autorepetición de la misma hasta que la suelte). Pulse DELETE (CAPS SHIFT y 0) para eliminar la P equivocada y luego entre " (SYMBOL SHIFT y P) para insertar el símbolo correcto. Observe que se coloca sin sobreimprimir nada de lo escrito. Si efectúa cualquier fallo al teclear, puede corregirlo de manera similar, recordando que, ante todo, debe eliminar lo escrito y luego imprimir las correcciones.

Ahora, al pulsar ENTER, el computador escribirá su mensaje (Hola, en este caso) en la parte superior de la pantalla o inmediatamente debajo del anterior, si todavía estaba allí.

Encontrará una detallada descripción del teclado en el Capítulo I del manual de programación en BASIC.

3º.- NUMEROS, LETRAS Y EL COMPUTADOR COMO CALCULADORA.

Hemos vistó ya la manera de ordenar al computador que imprima letras y gráficos en la pantalla empleando PRINT. También hemos visto que ENTER se emplea para ordenar al computador la ejecución del comando que acabamos de entrar. Desde ahora en adelante, por tanto no mencionaremos el empleo de ENTER cada vez que se emplea un comando, asumiendo que damos por descontado su uso al final de cada línea.

Los números son manejados por el computador con mayor facilidad que las letras. En el capítulo anterior, ya lo hemos insinuado al decir que el computador espera un número después de PRINT si no se emplean las comillas.

Así que si tecleamos

```
PRINT 2
```

el número 2 aparecerá en la pantalla.

Se pueden mezclar letras y números :

```
PRINT 2,"ABC"
```

Observe que hay un espacio vacío en la pantalla entre 2 y ABC.

Ahora entre

```
PRINT 2;"ABC"
```

y luego

```
PRINT 2"ABC"
```

Usando una coma entre partículas después de PRINT, las espacía 16 columnas; empleando punto y coma las imprime seguidas y no empleando nada el computador dá la señal de error.

La expresión PRINT puede emplearse junto con las funciones matemáticas del teclado. De hecho el ZX Spectrum puede emplearse como una calculadora electrónica.

Por ejemplo :

```
PRINT 2+2
```

La respuesta aparecerá en la pantalla. Compárela con la respuesta a

```
PRINT "2+2"
```

Es posible combinar ambas para algo más útil. Pruebe

```
PRINT "2+2=";2+2
```

Pruebe también otras expresiones aritméticas :

```
PRINT 3-2
PRINT 4/5
PRINT 12*2
```

El símbolo * se emplea como signo de multiplicar en lugar de X para evitar confusiones con la letra x; y / se emplea en lugar de ÷.

Experimente con diferentes cálculos. Si lo desea, puede emplear números negativos o decimales :

Si no tiene bastante con las 22 líneas de la parte superior de la pantalla, observará que algo interesante sucede ; toda la imagen se desplazará hacia arriba una línea y la superior, por tanto, se perderá. Esto se llama 'scrolling' (desplazamiento hacia arriba).

No siempre los cálculos se efectúan en el orden que cabría esperar. Como ejemplo, pruebe :

```
PRINT 2+3*5
```

En este caso, lo 'lógico' sería coger 2, añadirle 3 (que dá 5) y luego multiplicar por 5, con un total de 25. Las multiplicaciones, igual que las divisiones, se ejecutan antes que las sumas y restas, así que la expresión '2+3*5' en el computador significa 'tomar 3 y multiplicarlo por 5, dando 15 y luego añadirle 2, con un total de 17'. Y así se comprobará en pantalla, con la respuesta 17.

Como que las multiplicaciones y divisiones se operan primero, diremos que tienen prioridad más elevada que la suma y la resta. Respecto una de la otra, la multiplicación tiene la misma prioridad que la división , lo que significa que éstas se efectuarán en orden de izquierda a derecha, a medida que aparezcan en la expresión. Cuando estos cálculos se hayan efectuado, continuará con las sumas y restas que, entre sí, tienen la misma prioridad y se ejecutarán de manera igual a las anteriores, en orden de izquierda a derecha.

Veamos, como ejemplo, la forma en que el computador trabajaría con la expresión:

```
PRINT 20-2*9+4/2*3
```

```
1°)   20-2*9+4/2*3  }
2°)   20-18+4/2*3   }   Primero se realizan las multiplicaciones y
3°)   20-18+2*3     }   divisiones de izquierda a derecha
4°)   20-18+6       }
5°)   2+6           }   y luego las sumas y restas en el mismo orden.
6°)   8             }
```

A pesar de que todo lo que necesita para efectuar sus cálculos, es conocer si una operación tiene prioridad más elevada que otra, el computador realiza esto asignando un número de 1 al 16 a cada operación para representar su prioridad : * y / tienen prioridad 8 y + y -, prioridad 6.

Este orden de cálculo es absolutamente rígido, pero Vd. puede obviarlo con el empleo de paréntesis ; todo lo que esté encerrado entre paréntesis es evaluado en primer lugar y luego tratado como un simple número. Así pues,

```
PRINT 3*2+2
```

da el resultado 6+2=8, pero

```
PRINT 3*(2+2)
```

dará el total 3*4=12

A veces resulta útil introducir en el computador expresiones como las anteriores ya que, cuando el computador espera un número, puede introducirle una expresión en su lugar y entonces trabajará con el resultado de la misma. Las excepciones a esta regla son tan escasas que se especificarán individualmente en cada caso..

Puede entrar números decimales o con notaciones científicas como si fuera una calculadora normal. En éstos, trás un número ordinario (con o sin parte decimal), escribirá un exponente que consiste en la letra e y luego un número (que puede ser negativo). La parte exponente desplaza la coma decimal hacia la derecha (o hacia la izquierda si el exponente es negativo), es decir, multiplica (o divide) el

número original por 10 tantas veces como indique el número que acompaña a e. Por ejemplo,

2.34e0=2.34
2.34e3=2340
2.34e-2=0.0234 y así sucesivamente.

(Pruebe a escribir esto en el computador). Este es uno de los pocos casos en que no puede sustituir un número por una expresión : por ejemplo, no puede escribir

(1.34+1)e(6/2)

Puede también tener expresiones cuyos valores no sean números, sino cadenas de letras. Ya ha visto la forma simple de indicarlo con anterioridad : el grupo o cadena de letras encerrado entre comillas Es bastante similar a la forma más simple de expresión numérica. Lo que no ha visto todavía es el uso de + (suma) en las cadenas (no se emplea -,* ó / de forma que aquí no hay problemas con prioridades). Sumar cadenas simplemente significa añadir una detrás de la otra, así que pruebe;

PRINT "tor"+"o bravo"

Puede juntar tantas cadenas como quiera en una sola expresión y, si lo desea puede incluso emplear paréntesis.

4º- ALGUNOS COMANDOS SENCILLOS

La memoria del computador puede emplearse para guardar toda clase de cosas. Hemos visto, con anterioridad, que el comando PRINT nos permite imprimir letras, números y resultados de cálculos empleando letras y números en la pantalla.

Si queremos ordenar al computador que recuerde un número o una cadena de letras, entonces debemos dedicar una parte de la memoria para este fin.

La mayoría de las calculadoras de bolsillo tienen una tecla llamada 'memoria' que se emplea para 'recordar' números que se emplearán más adelante. Su computador puede hacer bastante más que eso : tiene tantas 'cajitas de memoria' como desee y, además puede asignarle un nombre a cada una.

A modo de ejemplo, suponga que desea 'memorizar' su edad. Se emplea, en este caso, el comando LET (asignar) que es el keyword de la letra L: supongamos que es 34 años.

LET edad=34

Lo que ocurre al emplear el comando LET es que una cierta parte de la memoria recibe el nombre 'edad' y guarda, en su interior, la cifra 34. Para recuperar esta información almacenada, entre :

PRINT edad

y obtendrá de nuevo el número 34. Es muy sencillo cambiar el contenido de la 'caja' llamada 'edad'. Entre :

LET edad=56

y a continuación

PRINT edad

y aparecerá 56 en la pantalla.'edad' es un ejemplo de variable, así llamada porqué su valor puede variar. Es posible combinar la impresión de un mensaje directo en la pantalla con el valor de una variable. Teclee :

PRINT "Su edad es ";edad

De todas formas, el computador tiene muchas más utilidades que el simple recordatorio de números con nombres asignados. Puede recordar también cadenas de letras. Para diferenciar entre variables numéricas y variables de cadena (como son llamadas), se emplea el símbolo del dólar ($) al final del nombre de la variable.

Por ejemplo, si queremos guardar la cadena de letras

"Su edad es "

le podemos asignar el nombre a$

(las variables de cadena deben tener una letra única seguida de $). Así que teclee :

LET a$="Su edad es "

Si ahora entra

PRINT a$

observará la cadena de letras impresa en la pantalla.

Si el computador no ha sido desconectado a lo largo de este capítulo, teclee :

```
PRINT a$;edad
```

y observe lo que resulta.

Hay otras maneras de introducir información al computador sin emplear el comando LET.

Por ejemplo el comando INPUT, en su expresión más simple, informa al computador de que debe esperar información desde el teclado. En lugar de teclear LET etc, cada vez, puede entrar

```
INPUT edad
```

Una vez haya pulsado ENTER aparecerá un cursor L parpadeante en la pantalla. Esto significa que el computador espera que Vd. le entre información. Así que entre su edad y luego pulse ENTER de nuevo. A pesar de que nada parece haber sucedido, la variable tiene ahora el valor que le ha entrado. Tecleando

```
PRINT edad
```

lo demostrará.

Podemos combinarlo todo junto en una serie de comandos.

Entre

```
LET b$="Cual es su edad ?"
LET a$="Su edad es "
INPUT (b$);edad:PRINT a$;edad
```

Observe que la última línea está formada por dos comandos separados por el símbolo dos puntos (:).

5°.- FUNDAMENTOS DE PROGRAMACION.

Hasta el presente hemos dicho al computador lo que debía hacer directamente desde el teclado. A pesar de poder combinar los comandos entre sí, sólo se pueden obtener limitadas aplicaciones con este método.

La gran cualidad de los computadores es que son programables. Esto significa que podemos darle una serie de instrucciones para que las ejecute secuencialmente.

Cada computador tiene su propio lenguaje que nos permite comunicarnos con él. Algunos lenguajes son extremadamente sencillos, de manera que el computador los entiende fácilmente. Por desgracia los lenguajes que son sencillos para el computador son complicados para los humanos. En algunos casos, lo contrario también se cumple, los lenguajes relativamente fáciles para nosotros, resultan difíciles para el computador de manera que hay que traducirlos o interpretarlos

El ZX Spectrum utiliza un lenguaje de alto nivel llamado BASIC.

La palabra BASIC es el anagrama de Beginners All-Purpose Symbolic Instruction Code (Algo así como Código de símbolos multi-uso para instrucción de principiantes) y fue inventado en el Darmouth College de New Hampshire, USA, en 1964. Es usado en la inmensa mayoría de computadores personales pero, a pesar de sus similitudes, existen sutiles diferencias entre cada uno de ellos. Esta es la razón por la que este manual se escribe específicamente para el ZX Spectrum. De todas maneras el BASIC del Spectrum es muy parecido al adoptado en general por la mayoría, de forma que no le resultará muy difícil adaptar cualquier programa en BASIC de otro computador al ZX Spectrum Al contrario que en otros BASICs el del ZX Spectrum no permite omitir el comando LET cuando se asignan valores a las variables.

Existe un límite para el número de instrucciones que pueden almacenarse en el computador. El ZX Spectrum indica este límite emitiendo un zumbido.

Cuando se programa en BASIC es necesario que el computador conozca el orden en que deseamos que ejecute las instrucciones. Por consiguiente cada línea de la secuencia de instrucciones debe tener un número precediéndola. Lo más corriente es empezar en 10 e incrementar este número de 10 en 10 a cada línea. Ello permite la inserción de líneas adicionales, si se olvidaron o si ha modificado el programa.

Veamos un programa sencillo. Emplea la serie de comandos mencionados en el capítulo anterior. Si deseamos repetir una serie de comandos sería necesario entrarlos cada vez. Un programa soluciona esta necesidad.

Entre lo siguiente, con ENTER al final de cada línea :

```
10 LET b$="Cual es su edad ?"
20 LET a$="Su edad es "
30 INPUT (b$);edad
40 PRINT a$;edad
```

Observe que no es necesario entrar espacios, excepto en el interior de las comillas.

No sucede nada apreciable hasta que le digamos al ordenador que empiece a trabajar. Esto se logra mediante RUN (el keyword de la R).

Entre dicho comando y observe lo que ocurre entonces:

Aparece en la parte inferior de la pantalla la frase

Cual es su edad ? L

con el cursor el L, esperando que le dé los datos por el teclado

Habrá observado también un cursor en forma de flecha al entrar cada línea. Este indica la última línea entrada. Si desea ver el programa de nuevo simplemente pulse ENTER de nuevo (o LIST).Puede emplear RUN para ejecutar el programa tantas veces como desee. Cuando se haya cansado de él, puede eliminarlo empleando el comando NEW. Este comando borra el programa almacenado en la memoria y le deja 'todo a punto' para introducir un nuevo programa.

Pulse NEW y, a continuación LIST y vea lo que sucede. (Aparecerá al pié de la pantalla el copyright de Sinclair)

Para resumir :

Cuando entra Vd. un comando precedido por un número, el computador no lo interpreta como un simple comando sino como una línea de programa. El computador no lo ejecuta sino que lo almacena para más adelante.

El ZX Spectrum, diligentemente, escribe en la pantalla (ó lista) todas las líneas del programa que ha entrado con un cursor en la última línea entrada.

El computador no ejecuta estas líneas inmediatamente sino que las almacena en su interior.

Para hacer que el computador ejecute dichas líneas debe emplear el comando RUN.

Siempre que pulse ENTER obtendrá el listado de nuevo.

Vamos a considerar otro programa sencillo. Este será algo más 'matemático' e imprimirá los cuadrados de todos los números entre el 1 y el 10 (el cuadrado de un número es el resultado de multiplicar dicho número por sí mismo).

Para generar números de 1 a 10 introducimos otro concepto de programación en BASIC. Este es el método por el cual hacemos contar al computador. Anteriormente hemos visto que los números pueden almacenarse en el computador asignándoles un nombre (ó, técnicamente, asignando un valor a una variable). Vamos a hacer que la variable x empiece con el valor 1 y lo incremente sucesivamente de 1 a 10. Esto se consigue empleando el comando FOR...TO...STEP (Para x...a...paso...)

Así, para entrar este programa pulse NEW para eliminar el anterior y teclee a continuación :

```
10 FOR x=1 TO 10 STEP 1 (Para x=1 a 10 paso 1)
```

Normalmente puede omitirse el STEP 1 (sólo se emplea para STEP (pasos) distintos de 1)

La siguiente línea debe decirle ahora al computador que tiene que hacer con x cualquiera que sea el valor que tenga, así que

```
20 PRINT x,x*x
```

Finalmente necesitamos una línea que le órdene al computador dar el siguiente valor a x, por tanto entre :

```
30 NEXT x        (NEXT significa próxima)
```

Al alcanzar esta instrucción el computador retrocede a la línea 10 y repite la secuencia. Cuando ha llegado al valor 10 de x el computador se dirige a la siguiente línea del programa, por ejemplo, la 40

El programa debe aparecer en pantalla de esta manera :

```
10 FOR x=1 TO 10 STEP 1
20 PRINT x,x*x
30 NEXT x
```

Para completarlo deberíamos colocar otra línea que dijera al computador que el programa ha terminado con el valor x=10, así que teclee:

```
40 STOP
```

Si ejecuta ahora el programa (mediante RUN), aparecerán dos columnas, la primera con los valores de x y la segunda con los de x*x o cuadrados de x. Si queremos encabezar dichas columnas, añadiremos otra línea, como ésta.

```
5 PRINT "x","x*x"
```

Observe que aunque ha entrado esta línea después de todas las anteriores, el computador la coloca en primer lugar al tener un número de línea inferior.

Pruebe de escribir programas empleando otras funciones matemáticas. Si tiene alguna duda en su utilización consulte las páginas necesarias del manual de programación BASIC.

6°.- EL EMPLEO DEL CASETTE

Es bastante engorroso tener que teclear programas en el computador cada vez que desea emplearlos. El ZX Spectrum tiene la posibilidad de grabar programas en cintas magnéticas empleando un casette normal. Si tiene un programa en la memoria del computador intente salvarlo con el siguiente procedimiento.

Si puede guardar programas en casette puede recuperarlos cuando le venga en gana.

La mayoría de casettes domésticos sirven para este cometido. En lo que al computador se refiere, el más humilde casette portátil es, como mínimo, tan bueno como el más costoso equipo estéreo y, al menos, no es tan complicado de manejar. Un contador de cinta es realmente útil en estos casos aunque no imprescindible ni mucho menos.

El casette debe tener un conector para micrófono y otro para auricular (MIC y EAR) respectivamente. Si este último no existe pruebe de conectarlo al conector de altavoz exterior. Estos conectores deben ser adecuados para jacks de 3,5 mm.(los más usados), ya que de lo contrario no dan una señal suficiente para el computador.

Cualquier cinta es válida aunque se recomienda usar las de bajo nivel de ruido (Low Noise). Una vez en posesión de un casette adecuado conéctelo al computador mediante los cables que se suministran. un cable debe conectar la salida de micrófono MIC del computador con la entrada correspondiente del casette y el otro la salida EAR del computador con la correspondiente al auricular o altavoz exterior del casette. (No hay peligro de dañar el computador si inadvertidamente conecta los cables al revés).

Cuando emplee el comando SAVE para guardar un programa en cinta, debe asegurarse de que el cable correspondiente a EAR está desconectado (en cualquiera de sus extremos). De lo contrario, no obtendrá más que una nota sostenida, que no le será de utilidad alguna.Esto sucede porqué el casette, cuando está grabando, amplifica la señal que le entra por el micro y, por consiguiente sale de esta forma por la salida de EAR (auricular) retornando al computador y realimentándose como un bucle, que oscila y, por tanto, enmascara la señal que trata de grabar

Entre algún programa en el computador, por ejemplo el programa de cuadrados del capítulo anterior, y luego teclee :

SAVE "cuadrados"

y el nombre 'cuadrados' es el que identifica el programa mientras está en la cinta. Puede elegir cualquier nombre que no exceda de 10 caracteres que pueden ser números o letras.

El computador contestará a lo anterior con un mensaje :Start tape then press any key (Ponga en marcha el casette y luego pulse cualquier tecla, sería la traducción aproximada) . Ante todo efectuaremos un simulacro de grabación para ver lo que sucede : no poner en marcha el casette, pero apretar una tecla del ZX Spectrum y vigilar el marco de la pantalla del televisor. Observará configuraciones de lineas horizontales de colores.

5 segundos de líneas rojas y azul claro, de cerca de 10 mm. de ancho y moviéndose lentamente hacia arriba.
Un breve destello de líneas azul y amarillas.
1 segundo de normalidad
2 segundos de nuevo de color rojo y azul pálido.
aprox. un segundo de líneas azules y amarillas de nuevo.

Pruébelo tantas veces como sea necesario para identificar la secuencia. La información es guardada en dos bloques y ambos tienen una señal inicial que corresponde a las líneas rojas y azul pálido y la información propiamente dicha que corresponde a las líneas azules y amarillas. El primer bloque contiene el nombre y varios bytes de información acerca del programa y el segundo bloque es el propio programa con todas las variables incluidas. La pausa sin rayas entre ambos bloques es simplemente una separación para diferenciarlos.

Ahora vamos a efectuar realmente la operación de grabar la señal en el casette.

1) Posicionar la cinta en algún lugar vacío o que desee grabar encima (borrando lo anterior, claro está).

2) Entrar

SAVE "cuadrados" (y ENTER)

3) Poner en marcha el casette (en posición de grabación)

4) Pulsar cualquier tecla del computador.

5) Observar la pantalla como antes. Cuando el computador ha terminado (con el informe # OK) detener el casette.

Para asegurarse de que todo ha ido bien, puede comprobar la señal de la cinta contrastándola con la del programa todavía en el computador mediante el comando VERIFY (Verificar).

1) Poner el volumen del casette a mitad de recorrido (si tiene control de tono, colocarlo a tope de agudos) y conectar el jack de EAR.

2) Rebobinar la cinta hasta un poco antes de empezar el programa.

3) Pulsar

VERIFY "cuadrados"

(VERIFY es en modo extendido, pulsando R)

4) Poner en marcha el casette en posición PLAY.

El contorno de la pantalla alternará entre rojo y azul claro hasta que la cinta encuentre el programa a verificar; entonces observará las mismas líneas que cuando estaba grabando. En la pause intermedia, se imprimirá en pantalla Program cuadrados. Cuando el computador está buscando algo en la cinta imprime el nombre de todo lo que encuentra en su búsqueda. Si, después del nombre deseado, observa las líneas mencionadas anteriormente y , una vez detenido el computador aparece en la pantalla el informe # OK, es señal de que su programa está bien guardado en cinta y, por tanto, puede saltarse las instrucciones que damos a continuación. En caso contrario, algo ha ido mal y es mejor que lea las siguientes instrucciones para determinar las causas del fracaso.

Normas para una correcta grabación de su programa

Ha salido el nombre ? (Al verificar)
En caso contrario o bien el programa no fué correctamente grabado en la primera operación o bien no ha sido convenientemente leido al verificar. Necesita saber cual de los dos casos es el correcto. Para saber si fué correctamente grabado, rebobine la cinta hasta un poco antes de empezar la grabación y oigala a través del altavoz del casette (seguramente tendrá que desconectar el jack de EAR para lograrlo). Las líneas rojas y azul claro dan una nota sostenida muy clara y aguda y la información (rayas amarillas y azules) proporciona un sonido bas-

tante desagradable, parecido a un mensaje en morse en medio de una ventisca. Ambos sonidos son bastante elevados de forma que a pleno volumen se hace difícil mantener una conversación.

Si no oye estos ruidos a través del altavoz, casi con seguridad que el programa no ha sido grabado. Compruebe los cables de cada enchufe que estén en posición correcta. Asegúrese de que el cable de MIC está conectado y el de EAR desconectado. Ocurre con algunos cassettes que el jack no hace contacto si está demasiado apretado. Trate de retirarlo un par de milímetros si observa que está en posición forzada. Compruebe también que no ha empezado la grabación en el trozo neutro que llevan las cintas en cada extremo. Cuando haya efectuado todas estas comprobaciones, pruebe SAVE de nuevo.

Si, por el contrario, oye los ruidos perfectamente por el altavoz la operación SAVE se ha desarrollado con normalidad y el fallo está en la lectura de la cinta.

Compruebe los cables de nuevo y también el nivel de volumen. Si está demasiado bajo,el computador no 'oirá' la señal correctamente; si es demasiado alto entonces la señal estará distorsionada. Debe de oirse a través del altavoz del computador. Hay una amplia gama de valores aceptables de forma que debe realizar varios intentos.

El caso siguiente se produce cuando el computador encuentra el programa y escribe su nombre pero el programa todavía no se carga. Hay varias posibilidades que son :

Se ha equivocado al teclear el nombre, o bien en SAVE (cuando el computador escribe el nombre erróneo en pantalla) o en VERIFY : el computador ignora el nombre y continua con sus líneas alternando rojo y azul pálido indefinidamente.

Hay un auténtico error en la cinta : el computador lo indicará con el informe R Tape loading error (R error en cinta), que significa, en este caso, que no ha logrado verificar el programa. Haga SAVE de nuevo.

También es posible que el nivel del volumen del casette no sea suficientemente alto; pero no puede ser demasiado grave puesto que ha 'leido' bien el primer bloque.

Ahora, supongamos que ha guardado el programa en cinta satisfactoriamente y verificado el mismo con éxito. Cargarlo de nuevo equivale a verificarlo sólo que ahora debe pulsar

LOAD "cuadrados"

en lugar de

VERIFY "cuadrados"

LOAD está en la tecla J. Dado que la verificación ha sido correcta no tendrá problemas en la carga.
LOAD borra el anterior programa (y variables) del computador antes de cargar el nuevo desde el casette.

Una vez el programa ha sido cargado, el comando RUN hará que éste se ejecute.
Es posible comprar programas pregrabados en casette. Deben ser específicamente dedicados al ZX Spectrum : diferentes tipos de computador tienen distinta forma de grabar programas, de forma que no pueden usarse cintas de otros computadores.
Si su cinta tiene más de un programa grabado en la misma cara, puede darle a cada uno un nombre y puede escoger el que desea cargar con el comando LOAD : por ejemplo, si el que desea se llama 'androide'

debe entrar

LOAD "androide"

Si escribe LOAD "" significa LOAD el primer programa que encuentre, lo que es muy útil cuando no recuerde el nombre del programa que busca. (N.del T.) Se recomienda no emplear mayúsculas y minúsculas mezcladas en un mismo nombre, p.e. empezar el nombre con letra mayúscula puede ser correcto en lenguaje normal pero puede causar problemas a la hora de recordar el nombre).

7°.- COLORES

Una de las características principales que inducen a comprar el ZX Spectrum es su capacidad de utilizar colores en la pantalla del televisor. Dicha pantalla se divide en dos sectores. La parte exterior es llamada BORDER (marco, contorno, etc) y el área central PAPER (papel). Es posible cambiar los colores de estas áreas a voluntad, mediante el teclado o en medio de un programa.

El ZX Spectrum puede manejar ocho colores a los que se les asigna un número de 0 a 7. A pesar de que esta asignación pueda parecer aleatoria, de hecho está pensada para que dé tonos decrecientes de gris en un televisor en blanco y negro.

He aquí una relación de los mismos como referencia ; también se encuentran escritos encima de la tecla correspondiente en el teclado.

0 Negro
1 Azul
2 Rojo
3 Magenta (o violeta claro)
4 Verde
5 Cian (azul claro)
6 Amarillo
7 Blanco

La primera vez que se conecta el computador, el sistema trabaja en blanco y negro. Así que el valor normal de BORDER y PAPER es el 7, es decir, blanco. El color de los caracteres que aparezcan en la pantalla vendrá definido por el comando INK (tinta, texto). Inicialmente INK vale 0, es decir, negro. Estos tres valores los establece el computador cada vez que se pone en marcha.

No obstante, puede cambiar estos valores. Por ejemplo, teclee :

```
BORDER 2      (y ENTER, naturalmente)
```

observará que el contorno cambia ahora de blanco a rojo. Esto incluye el área de la parte inferior donde se insertan comandos e instrucciones. Pruebe de entrar diferentes números y verá como va cambiando el color.

A continuación intente cambiar el área del centro de la pantalla mediante

```
PAPER 5
```

El comando PAPER se obtiene en modo extendido como ya se mencionó en capítulos anteriores. Se consigue pulsando CAPS SHIFT y SYMBOL SHIFT al mismo tiempo y, a continuación, manteniendo una pulsada, apretar la tecla C. Cuando la tecla ENTER haya sido pulsada dos veces el centro de la pantalla cambiará a azul pálido. El primer ENTER cancela el comando PAPER todavía almacenado en el computador, pero sólo cuando se aprieta ENTER por segunda vez (causando la aparición de cualquier listado que hubiera en el computador), aparece el color de PAPER deseado. Si está empleando un televisor de color y éste no ha cambiado, pruebe de ajustar el control de color del televisor e incluso, el mando de sintonía.

El comando INK es similar al PAPER y controla el color de los caracteres que aparecen en el área de PAPER de la pantalla. Naturalmente, si INK y PAPER tienen el mismo color no se distinguirá nada

en la pantalla !.

Los comandos BORDER, PAPER e INK pueden emplearse en los programas. Aquí tenemos uno que muestra la gama de colores disponibles.

```
10 FOR x=0 TO 7
20 BORDER x
30 PAPER 7-x:CLS
40 PAUSE 50
50 NEXT x
```

Este programa, cuando lo ejecute, mostrará los ocho colores contrastando los de PAPER y BORDER. El comando CLS después de PAPER ordena al computador borrar lo que tenga en pantalla y emplear el nuevo color de PAPER. El comando PAUSE (pausa) detiene el programa durante 1 segundo para que veamos lo que está ocurriendo (Intente ejecutar el programa sin la línea 40). Para observar como trabaja el comando INK, teclee el siguiente programa, después de emplear el comando NEW.(que elimina el programa anterior).

```
10 BORDER 7
20 PAPER 1
30 INK 4
40 PRINT "Caracteres verdes sobre fondo azul"
```

Existen otros comandos relacionados con el empleo de colores en el ZX Spectrum que serán detallados en el manual de programación en BASIC.

8°.- SONIDO

El ZX Spectrum puede generar una infinita variedad de sonidos. La frecuencia de la nota y su duración puede regularlas el usuario. El comando BEEP se emplea para ordenar al computador que genere un sonido. El comando BEEP se obtiene en modo extendido y la tecla Z.

La frecuencia 'central' del comando BEEP es la nota DO. Dicha frecuencia puede variarse con el comando BEEP y puede obtenerse cualquier nota expresándola en semitonos o partes de semitono arriba o abajo de la frecuencia central. Si se entra el comando

```
BEEP 2,0
```

el computador emitirá un DO durante dos segundos.

Los dos números que siguen a BEEP controlan la clase de nota emitida, el primero da la duración de la nota en segundos y el segundo el tono de la nota en semitonos por encima de DO. Entonces el código del tono de DO es 0, el del DO sostenido es 1, el RE es 2 y así sucesivamente hasta el siguiente DO de la octava superior que será 12, ya que doce semitonos hacen una octava. Puede continuar con 13 y números mayores si quiere, cuanto más alto el número más elevado será el tono obtenido.

Pruebe lo siguiente :

```
BEEP 1,4:BEEP 1,2:BEEP 2,0
```

y oirá las tres primeras notas de una popular melodía. Dado que se pueden combinar bastantes BEEPs con los dos puntos de separación igual que se ha indicado podrá, con paciencia,producir una melodía entera. Pruebe algo más que las tres notas anteriores.

(Los dos puntos (:) no sólo unen comandos BEEP sino que puede emplearlos para construir comandos compuestos con cualquiera de los comandos simples).

Como un ejemplo algo más complicado, puede improvisar un efecto de 'camaleón sonoro' mezclando los comandos BEEP y BORDER :

```
BORDER 1:BEEP 1,14:BORDER 3:BEEP 1,16:BORDER 4:BEEP 1,12:
BORDER 6: BEEP 1,0:BORDER 5:BEEP 4,7:BORDER 1
```

(No se preocupe por el hecho de ocupar más de una línea en la pantalla. El computador ni se entera).

Un sencillo programa para emitir una serie completa de notas sería:

```
10 FOR x=0 TO 24
20 BEEP 2,x
30 NEXT x
```

Hay una infinidad de cosas que pueden hacerse con este comando. En el manual de Programación se verá con mayor detalle.

Para notas más bajas del DO central, el número de semitonos debe indicarse en negativo (precedido del signo -).

9°.- QUE HAY EN EL INTERIOR ?

La figura de la página 29 del manual de introducción (pequeño) en idioma inglés muestra la disposición interior de los elementos que componen el ZX Spectrum. (N.del T. Esta foto corresponde a las primeras series del computador. En la versión actual, existen los mismos elementos pero con distinta disposición de montaje).

Como podrá observar casi todo se denomina con abreviaciones de tres letras. Las piezas rectangulares de plástico negro con patitas son los circuitos integrados que realmente efectúan todo el trabajo. En el interior de cada uno de ellos hay una pequeña pastilla de silicio de sólo 6x6 mm. conectada con finos cables a las patitas exteriores. En cada 'chip' (pastilla) de silicio existen miles de transistores que forman los circuitos electrónicos que integran el computador.

El 'cerebro' que dirige la operación es el microprocesador, llamado normalmente CPU (Central Processor Unit= Unidad central de proceso). En este caso se trata del Z80 A, versión más rápida del popular Z80.

El microprocesador controla el computador, realiza los cálculos, toma nota de cuales teclas se han pulsado, decide lo que debe hacer en consecuencia y, en general controla todo lo que debe hacer el computador. De todas formas, a pesar de su 'inteligencia' no podría hacer nada por sí solo. No sabe nada de BASIC ni comas decimales, por ejemplo, y debe obtener todas sus instrucciones de otro chip, la ROM (Read Only Memory= Memoria de sólo lectura). La ROM contiene una larga lista de instrucciones que forman un programa de computador, que indican al procesador lo que debe hacer en cualquier previsible circunstancia. Dicho programa no está escrito en BASIC sino en lo que se llama 'código máquina' del Z80 y adopta la forma de una larga secuencia de números. Hay 16384 números de los citados (16*1024), por lo que se dice que la ROM del ZX Spectrum es de 16 K de BASIC (1 K es igual a 1024 números, o bytes).

A pesar de que hay otros computadores con el mismo procesador, la secuencia de instrucciones es exclusiva del ZX Spectrum y ha sido escrita especialmente para él.

Los ocho circuitos paralelos (de menor tamaño) se usan para la memoria. Es llamada memoria RAM (Random Access Memory= Memoria de acceso aleatorio) y existen dos otros circuitos integrados que trabajan conjuntamente con ella. La RAM es el lugar en que el computador almacena la información que tiene que guardar, los programas en BASIC, las variables, la imagen de la pantalla y varios otros elementos que mantienen al computador operativo.

El integrado grande que queda es el SCL (Sinclair Computer Logic). Este actúa como 'centro de comunicaciones', proporcionando al procesador toda la ayuda que requiera ; también 'lee' la memoria para ver en que consiste la imagen del televisor y mandar las apropiadas señales a través del correspondiente interface de TV.

El codificador PAL es un conjunto de componentes que convierten la salida lógica del chip anterior en señales adecuadas para el televisor en color.. El regulador convierte le tensión de entrada del alimentador en 5 Volt constantes

Esto concluye este manual de introducción. Si cree que lo ha asimilado completamente, entonces ya es hora de que se atreva con el manual de programación En BASIC. Buena suerte !.

CAPITULO I.- INTRODUCCIÓN

Tanto si leyó en primer lugar el fascículo de introducción como si ha empezado por éste, Vd. debe saber que los comandos tienen una ejecución inmediata y que las instrucciones deben llevar un numero en su encabezamiento y se almacenan para au ejecución posterior . Al mismo tiempo, conocerá ya los comandos PRINT, LET e INPUT, de uso general en el lenguaje BASIC y BORDER, PAPER y BEEP (de especial aplicación en el BASIC que emplea el Spectrum.

Este manual de BASIC reitera, con más profundidad, algunos de los conceptos encontrados en el fascículo de introducción. También encontrará unos ejercicios al final de cada capítulo. Procure efectuarlos pues muchos de ellos amplían algunos puntos superficialmente explicados en el texto.

Siempre que tenga una duda respecto al comportamiento del ordenador ante determinada instrucción, no dude en introducirla por el teclado. Es prácticamente imposible que pueda ocasionar algún desperfecto y no hay mejor explicación que observar lo que sucede. Así pues cuantos más programas propios experimente, mejor comprenderá el funcionamiento del Spectrum.

EL TECLADO.-

Los caracteres del ZX Spectrum incluyen además de los símbolos simples (letras, números, etc.), los comandos e instrucciones, con una simple pulsación de la tecla. Para conseguir todas estas funciones, determinadas teclas tienen cinco o más significados distintos, logrados por combinaciones de teclas (CAPS SHIFT ó SYMBOL SHIFT) y por los propios modos operativos del Spectrum.

El modo operativo viene indicado por el cursor que es una letra parpadeante que indica la posición donde se insertará lo que se escribe en el teclado.

El modo K significa que la máquina está esperando una instrucción o un número de línea. Esta posición es Al principio de la línea o después de THEN ó de :.

El modo L suele presentarse en todos los demás casos.

En los modos K y L, SYMBOL SHIFT y una tecla, proporcionarán el carácter escrito en rojo en esta tecla. CAPS SHIFT y una tecla proporcionará la función de control escrita en blanco en dicha tecla. CAPS SHIFT con las demás teclas no afecta a las palabras clave en el modo K y, en el modo L, convierte las mínúsculas en mayúsculas.

El modo C es una variante del modo L y todas las letras aparecen mayúsculas. CAPS LOCK produce un cambio del modo L al C y viceversa.

El modo E se utiliza para obtener caracteres adicionales. Tiene lugar después de que se pulsen simultáneamente las teclas de SYMBOL SHIFT y CAPS SHIFT y dura solamente una letra. Las dos teclas mencionadas juntas dan el comando superior en verde y, manteniendo una de las dos apretadas dan el comando inferior en rojo.

El modo G se obtiene pulsando simultáneamente CAPS SHIFT y 9 (GRAPHICS) y permanece hasta que se repita la operación a la inversa o se pulse simplemente 9. Una tecla de dígito proporcionará el carácter gráfico indicado (menos GRAPICHS o DELETE) y cada tecla de letras (menos V,W,X,Y ó Z), dará un gráfico definido por el usuario.

Manteniendo cualquier tecla apretada más de 2 ó 3 seg., comienza la repetición de la misma.

La introducción por el teclado se presenta en la mitad inferior de la pantalla a medida que se teclea, insertándose cada carácter inmediatamente antes del cursor. Este puede desplazarse a la izquierda o a la derecha con los cursores de dirección (CAPS SHIFT 5 y 8). El carácter precedente al cursor se borra con DELETE (CAPS SHIFT y 0). La línea completa puede borrarse entrando EDIT (CAPS SHIFT y 1) y , a continuación, ENTER.

Cuando se pulsa ENTER, se ejecuta la línea, se introduce en el programa o se utiliza como entrada de datos, cuando convenga, a menos que se produzca un error de sintáxis en cuyo caso aparece un ? parpadeante junto al error.

A medida que se entran las líneas del programa, se observa el listado apareciendo en la mitad superior de la pantalla. La última línea introducida se denomina línea en curso y se indica con el símbolo >; éste puede desplazarse con el empleo de las teclas de dirección (CAPS SHIFT 6 y 7) Si se pulsa EDIT, la línea en curso se duplica en la parte inferior de la pantalla y puede corregirse, sustituyendo a la original una vez corregida simplemente pulsando ENTER.

Cuando se ejecuta un comando o un programa, el resultado se visualiza en la mitad superior de la pantalla y se mantiene allí hasta que se introduce una línea de programa o se pulse ENTER con una línea vacía o se pulse SHIFT 6 ó 7. En la parte inferior aparece un informe que da un código al que se hace referencia en el apéndice B. El informe aparece en pantalla hasta que se pulse una tecla cualquiera (entrando en modo K)

En determinadas circunstancias, CAPS SHIFT junto con SPACE actúa como BREAK, deteniendo el ordenador con informe D o L. Ello se logra :

al final de una sentencia mientras se ejecuta un programa o

mientras el computador está empleando la impresora o el casette

LA PANTALLA.-

Tiene 24 líneas, con 32 caracteres cada una y está dividida en dos partes La superior comprende, como máximo, 22 líneas y visualiza el listado o la ejecución del programa Cuando la impresión en la parte superior, alcanza la parte inferior, se desplaza hacia arriba (scroll) una línea , si ello implicara la pérdida de una línea que no se hubiera visualizado todavía, entonces el ordenador se detiene con el mensaje SCROLL ? Si pulsa las teclas N, SPACE o STOP se hará que se interrumpa el programa con el informe D BREAK CONT repeats ; cualquier otra tecla permitirá que continúe el scrolling (ó desplazamiento hacia arriba). La parte inferior se emplea para la introducción de comandos, líneas de programa o entrada de datos y, también, para visualizar informes que dá el ordenador la parte inferior ocupa, en principio las dos últimas líneas pero se amplía para admitir lo que pueda entrar desde el teclado. Cuando llegue al espacio destinado al listado en la parte superior, dicha parte superior irá desplazándose hacia arriba para dejar sitio.

CAPITULO II.- CONCEPTOS DE PROGRAMACION EN BASIC

Resumen : Programas - Números de línea -
Edición correctora de programas con el empleo de ↓, ↑ y EDIT.
RUN, LIST, GO TO, CONTINUE, INPUT, NEW, REM, PRINT.
STOP en entrada de datos
BREAK

Teclee estas dos líneas de un programa de ordenador para obtener el resultado de la suma de dos números :

```
20 PRINT a
10 LET a=10
```

de modo que en la pantalla aparezca :

```
10 > LET a=10
20   PRINT a
```

Como observa, debido a que estas líneas comenzaban con números, no se "obedecieron" inmediatamente sino que se almacenaron como líneas de programa. También puede observar que no se almacenaron en el órden entrado aunque el símbolo > aparece en el número 10 que es la última instrucción entrada.

Teclee ahora a continuación :

```
15 LET b=15
```

y veamos lo que ocurre. Hubiera sido imposible insertar esta línea entre las dos anteriores si aquellas se hubieran numerado 1 y 2 en lugar de 10 y 20 (los números de línea tienen que ser enteros entre 1 y 9999). Por ello es recomendable, cuando se entra un programa en el ordenador, dejar espacios entre los números de línea para posibles posteriores modificaciones.

Ahora debe cambiar la línea 20 a :

```
20 PRINT a+b
```

Podría introducir la nueva instrucción por completo, pero resulta más fácil emplear el modo EDIT descrito en el capítulo de introducción. El > en la línea 15 se denomina el cursor del programa y la línea a la que apunta, línea en curso. Puede emplear las teclas ↓ o ↑ para desplazar el cursor del programa hacia abajo o hacia arriba (intente dejarlo en la línea 20).

Pulsando la tecla EDIT (CAPS SHIFT y 1), una copia de la línea en curso, en este caso la 20, se visualizará en la parte inferior de la pantalla. Mantenga oprimida la tecla → hasta que el cursor L se desplace al final de la línea y entonces, teclee :

```
+ b     (sin ENTER)
```

La línea en la parte inferior de la pantalla aparece ahora :

```
20 PRINT a+b
```

Pulse ENTER y sustituirá automáticamente la anterior línea 20, de modo que en la pantalla aparecerá :

```
10 LET a=10
15 LET b=15
20>PRINT a+b
```

Ejecute este programa con el empleo de RUN y ENTER y se visualizará la suma. Vuelva a ejecutar el programa y luego teclee :

```
PRINT a,b
```

Las variables siguen estando allí, aún cuando el programa haya acabado.

Hay un método útil empleando EDIT para prescindir de la parte inferior de la pantalla. Teclee cualquier cosa que carezca de significado (sin ENTER), e imagine que desea hacerlo desaparecer por carecer de valor. Una forma de lograrlo es pulsar la tecla DELETE . Otra forma más rápida de efectuarlo es la siguiente : Pulse EDIT y lo que se desea borrar desaparecerá para ser sustituido por una copia de la línea en curso. A continuación, pulse ENTER y dicha línea en curso volverá intacta al programa, dejando limpia la parte inferior de la pantalla.

Si introduce una línea defectuosa, por ejemplo :

```
12 LET b=8
```

ésta se introducirá en el programa y se lo perturbará. Para borrarla, simplemente teclee :

```
12      ( y ENTER, por supuesto)
```

Observará que la línea ha desaparecido y, con ella, el cursor del programa. Pulsando ↑ aparecerá en la línea 10 y pulsando ↓ aparecerá en la línea 15.

Teclee :

```
12      ( y ENTER)
```

De nuevo, el cursor del programa se ocultafá entre las líneas 10 y 15 Ahora pulse EDIT y descenderá la línea 15. Siempre que el cursor esté oculto entre dos líneas, EDIT bajará la siguiente línea después de la que teóricamente contiene al cursor. Teclee ENTER para limpiar la parte inferior de la pantalla.

Ahora teclee :

```
30      ( y ENTER)
```

Esta vez el cursor del programa está oculto tras el final del programa y, si pulsa EDIT, bajará la línea 20.

Finalmente, teclee :

```
LIST 15
```

Observará ahora en pantalla :

```
15 LET b=5
20 PRINT a+b
```

La línea 10 ha desaparecido de la pantalla, pero sigue estando en el programa (lo que puede comprobar pulsando ENTER), Los únicos efectos de LIST 15 son producir un listado a partir de la instrucción 15. Si tiene un programa muy largo, entonces LIST es una más útil manera de desplazar el cursor del programa en lugar de ↓ y ↑ .

LIST, sin ningún número, lista el programa a partir de su principio.

El efecto del comando NEW es borrar tanto los programas como las variables que hubiese almacenados en el Spectrum. A continuación, introduzca este programa que cambia las temperaturas de Fahrenheit a ºC.

```
10 REM conversión de temperaturas
20 PRINT "grados F","grados C"
30 PRINT
40 INPUT "Introducir grados F",F
50 PRINT F,(F-32)*5/9
60 GO TO 40
```

Necesitará teclear las palabras contenidas en la línea 10. Aunque GO TO tiene un espacio entre ambos vocablos es, en realidad, un solo comando (tecla G).

Ahora haga la ejecución correspondiente. Verá los encabezamientos impresos en la pantalla por la línea 20, mientras que la 10 es ignorada.En realidad el comando REM es puramente una referencia para el programador, ignorándolo el ordenador así como todo lo que haya escrito a continuación.

De momento, el ordenador ha llegado al comando INPUT en la línea 40 y está a la espera de que se introduzca, a través del teclado, un valor para la variable F (El cursor L lo indica claramente). Introduzca un número seguido de ENTER. Ahora, el ordenador presentará el resultado y está esperando otro número. Esto es así porqué la línea 60, GO TO 40 significa

que el ordenador, en lugar de ejecutar el programa y detenerse, ha de volver a la línea 40 y comenzar de nuevo. Por consiguiente, introduzca una nueva temperatura.

Después de introducir unos cuantos valores más, se preguntará si la máquina no se "aburre" de hacer siempre lo mismo. En realidad no es así, pero la siguiente vez que le pida un número puede teclear STOP. Entonces el ordenador presentará el informe D STOP in INPUT in LINE 30:1, que le indica por qué se ha detenido y en donde lo ha hecho (en el primer comando de la línea 30).

Si quiere continuar el programa, entre :

CONTINUE

y el ordenador le pedirá otro número.

Cuando se utiliza CONTINUE,el ordenador recuerda el número de línea del último informe que transmitió, a menos que fuera 0 OK, y salta de nuevo a ella. En nuestro caso la línea 40, el comando INPUT.

Sustituya la línea 60 por GO TO 31 y no observará ninguna diferencia en el desarrollo del programa. Si el número de línea de un GO TO se refiere a una línea no existente en el programa entonces el salto se dirige a la siguiente. Lo mismo es válido para RUN ; de hecho RUN, en si mismo, significa en realidad RUN 0.

Ahora, intróduzca mediante el teclado números hasta que la pantalla comience a llenarse. Cuando lo esté, el ordenador desplazará el conjunto de la parte superior de la pantalla una línea hacia arriba para dejar sitio, perdiendo el encabezamiento al salir por la parte superior. Este desplazamiento se llama "scrolling".

Cuando se canse de realizar esta operación, detenga el programa con el comando Stop y consiga el listado pulsando ENTER.

Examine la sentencia PRINT en la línea 50 . La puntuación en esta sentencia (la coma en este caso) es muy importante y sigue unas reglas diferentes a las clásicas habituales en gramática.

Las comas se utilizan para hacer que la impresión comience en el margen izquierdo , o en la parte central de la pantalla, dependiendo de lo que venga a continuación. Así en la línea 50, la coma hace que la temperatura en °centigrados, se imprima en medio de la línea. Por el contrario, con punto y coma, el siguiente número o cadena se imprimirá inmediatamente después de lo que le precede. Puede comprobarlo en la línea 50, si cambia la coma por un punto y coma.

Otro signo de puntuación que puede utilizar como tal en elos comandos PRINT es el apóstrofe ('). Este signo hace que lo que se imprima a continuación aparezca al principio de la siguiente línea en la pantalla, pero ello sucede, de todos modos, al final de cada comando PRINT, por lo que no tendrá mucha necesidad del apóstrofe. Esta es la razón por la que el comando PRINT en la línea 50 comienza siempre su impresión en una línea nueva y también por la que el PRINT de la línea 30 produce una línea vacía.

Si quiére inhibir lo anterior, de modo que después de un comando PRINT, lo siguiente quede en la misma línea, puede poner una coma, o un punto y coma, al final del primer comando. Para comprobarlo, sustituya la línea 50 sucesivamente por cada una de las sentencias :

```
50 PRINT F,
50 PRINT F;
```

y

```
50 PRINT F
```

y ejecute cada variante. Para completarlo, pruebe también con :

```
50 PRINT F'
```

El comando de la coma extiende todo en dos columnas, con el punto y coma se presenta todo junto, sin ninguno de los dos signos anteriores asigna una línea a cada número y lo mismo sucede con el apóstrofe (el apóstrofe dá una nueva línea por si mismo pero inhibe la automática).

Tenga presente la diferencia entre las comas y el punto y coma en los comandos PRINT ; además no debe confundirlos con los dos puntos (:) que se utilizan para separar comandos en una misma línea.

Ahora, teclee estas líneas suplementarias :

```
100 REM Este educado programa recuerda su nombre
110 INPUT n$
120 PRINT "Hola ";n$;" !"
130 GO TO 110
```

Nota . Deje un espacio al final de Hola y al principio de ! para mejorar la impresión.

Este es un programa independiente del anterior, pero puede mantener a ambos en el ordenador al mismo tiempo. Para ejecutar el nuevo programa, entre :

```
RUN 100
```

Puesto que este programa espera que se introduzca una cadena en lugar de un número, imprime dos comillas. Esto le ayuda a saber qué debe introducir. Pruébelo escribiendo cualquier cosa.

Otra vez volverá a encontrarse con dos comillas, pero no tiene que utilizarlas si no lo necesita. Pruebe esto por ejemplo: suprima las comillas (con → y DELETE) y teclee :

```
n$
```

Puesto que no hay comillas de cadena, el ordenador sabe que ha de realizar algún cálculo ; en este caso, el cálculo consiste en determinar el valor de la variable de cadena denominada n$, que es cualquier nombre que casualmente haya introducido en el teclado en la última ocasión. Por supuesto, la sentencia INPUT actúa como LET n$=n$, por lo que se mantiene invariable el valor de n$.

A continuación, con fines comparativos, teclee :

```
n$
```

de nuevo, pero esta vez sin suprimir las comillas de cadena. Ahora, precisamente para confundirle, la variable n$ tiene el valor "n$".

Si desea utilizar STOP para la introducción de datos, debe desplazar el cursor haciéndolo retroceder al principio de la línea, con el empleo de la tecla ← .

Volvamos de nuevo a este RUN 100 que teníamos para saltar a la línea 100. Podemos poner GO TO 100 en su lugar ?. Efectivamente, pero hay una pequeña diferencia. Haciendo RUN 100 suprime todas las variables y borra la pantalla mientras que GO TO 100 no borra nada. Pueden darse casos en que interese ejecutar un programa sin borrar ninguna variable. En este caso, entrar RUN podría tener efectos desastrosos. Por ello es preferible dejar la costumbre de teclear RUN por sistema para ejecutar un programa.

Otra diferencia es que puede teclear RUN sin número de línea y el ordenador siempre comenzará en la primera mientras que GO TO debe ir siempre acompañado de un número de línea.

Ambos programas se interrumpieron porqué tecleó STOP en la línea de entrada ; a veces por error, escribe un programa que no puede detener y que no se parará por si mismo. Teclee :

```
200 GO TO 200
RUN 200
```

Este programa tiene el aspecto de proseguir indefinidamente a no ser que se decida a desenchufar el aparato, pero hay un remedio menos drástico. Pulse CAPS SHIFT junto con la tecla SPACE, que tiene escrito BREAK por encima. El programa se interrumpirá con el informe L BREAK into program (interrupción del programa por ruptura).

Al final de cada sentencia, el programa comprueba si están pulsadas estas teclas y si lo están el ordenador se para. La tecla BREAK puede emplearse también cuando se utilice el casette, la impresora o cualquier otro dispositivo conectado al Spectrum.

En estos casos, hay un informe diferente D BREAK-CONT repeats. En

tales circunstancias, CONTINUE repite (y, de hecho, en la mayoría de los casos) la sentencia en donde se detuvo el programa ; pero después del informe L BREAK into program, CONTINUE pasa directamente a la siguiente sentencia después de permitir que se hagan algunos saltos.

Vuelva a ejecutar el programa del nombre y cuando el ordenador le pida que entre lo solicitado, teclee :

n$ (después de quitar las comillas)

n$ es una variable no definida y obtiene un informe de error 2 Variable not Found (variable no encontrada). Si ahora entra ;

LET n$="algo determinado"

(que tiene su propio informe de Ø OK, Ø:1) y

CONTINUE

encontrará que puede utilizar n$ como introducción de datos sin más problemas.

En este caso, CONTINUE realiza un salto al comando INPUT de la línea 11Ø. No hace caso del informe de la sentencia LET, porqué dijo OK y salta al comando al que se hizo referencia en el anterior informe, el primero en la línea 11Ø. Si un programa se interrumpe por algún error, en lugar de empezar de nuevo la ejecución puede resultar más sencillo rectificar la anomalía y emplear el comando CONTINUE.

Como se dijo anteriormente, el informe L BREAK into program es especial porqué después del mismo, CONTINUE no repite el comando en donde se detuvo el programa..

Los listados automáticos (los que no son resultado de un comando LIST sino que se producen después de introducir una nueva línea), quizás le desconcierten. Si introduce por el teclado en un programa con 5Ø líneas todas las sentencias REM

```
 1 REM
 2 REM
 3 REM
 .
 .
49 REM
5Ø REM
```

entonces será capaz de experimentar.

Lo primero a recordar es que la línea en curso (con ▶) siempre aparecerá en la pantalla y normalmente cerca de la parte central.

Pulse :

LIST (y ENTER, por supuesto)

y cuando el ordenador le pregunte scroll ? (porqué ha llenado la pantalla) pulse n para indicar "NO". El ordenador emitirá el informe D BREAK-CONT repeats como si hubiera tecleado BREAK. En alguna ocasión, pruebe lo que sucede si pulsa y en lugar de n ; n, SPACE y STOP cuentan como "no", mientras que todo lo demás cuenta como "si".

Ahora, vuelva a pulsar ENTER para obtener un listado automático. Deberá ver las líneas 1 a 22 en pantalla. Ahora , teclee :

23 REM

obtendrá las líneas 2 a 23 en la pantalla ; teclee :

28 REM

y obtendrá las líneas 27 a 48. (en ambos casos, al teclear una nueva línea, ha desplazado el cursor del programa y por ello se ha realizado un nuevo listado).

Quizás esto le parezca algo arbitrario. Realmente, trata de proporcionarle exactamente lo que necesita, aunque al ser los "humanos criaturas imprevisibles", no siempre el ordenador adivina correctamente.

El ordenador mantiene un registro no solamente de la línea en curso (la que ha de aparecer en pantalla), sino también de la línea superior de la pantalla. Cuando intenta hacer un listado, lo primero que hace es comparar la línea superior con la línea en curso.

Si la línea superior viene detrás, entonces no resulta oportuno comenzar en este punto y por ello el ordenador utiliza la línea en curso para una nueva línea superior y realiza su listado.

De cualquier otro modo, su método es comenzar la ejecución del listado a partir de la línea superior y proseguir hasta que la línea en curso sea objeto de listado, con un desplazamiento hacia arriba (scrolling) en caso de necesidad. Sin embargo, primero efectúa un cálculo aproximado para ver cuanto ocupa y si la respuesta es que tiene una magnitud excesiva, entonces el ordenador desplaza la línea superior hacia abajo para estar mucho más próxima a la línea en curso. Ahora, al haber determinado su línea superior, el ordenador comienza el listado a partir de la misma. Si, cuando llega al final del programa o al fondo de la pantalla, se ha listado la línea en curso, entonces se detiene el ordenador. De no ser así, se producirá un desplazamiento hacia arriba hasta que la línea en curso esté en la pantalla y por cada línea adicional que sea objeto de listado, el ordenador desplaza la línea superior hacia abajo en una línea, de modo que la línea superior se desplace a las proximidades de la línea en curso.

Experimente con estos desplazamientos, tecleando :

número de línea REM

LIST desplaza la línea en curso pero no la línea superior, por lo que los subsiguientes listados podrían ser distintos. Por ejemplo, teclee :

LIST

para conseguir el listado y luego pulse de nuevo ENTER para hacer que la línea Ø sea la línea superior. Debe tener las líneas 1 a 22 en la pantalla. Teclee :

LIST 22

que le proporcionará las líneas 22 a 43 ; cuando pulse ENTER de nuevo, vuelva a las líneas 1 a 22. Ello tiende a ser más útil para programas cortos que en programas largos.

Utilizando el programa lleno de sentencias REM anterior, entre :

LIST

y luego n cuando el ordenador le pregunte scroll?. Ahora, teclee :

CONTINUE

En este caso, CONTINUE tiene un carácter un poco "caprichoso" porqué se pone en blanco la parte inferior de la pantalla ; pero puede restaurar la normalidad con BREAK. La razón es que LIST era el primer comando en la línea, por lo que CONTINUE repite este comando. Lamentablemente, el primer comando en la línea es ahora CONTINUE, propiamente dicho, por lo que el ordenador se estabiliza realizando la función CONTINUE una y otra vez hasta que lo detenga.

Puede variar esta circunstancia sustituyendo LIST por

:LIST

para lo que CONTINUE da Ø OK(porqué CONTINUE salta al segundo comando en la línea el cual se toma como su final) ó

:: LIST

para lo que CONTINUE da N Statement lost(sentencia perdida)(porqué CONTINUE salta al tercer comando de la línea, el cual ya no existe).

Ya ha visto las sentencias PRINT, LET, INPUT, RUN ; LIST, GO TO, CONTINUE, NEW y REM y todas ellas pueden utilizarse como comandos directos o en líneas de programa.Ello es cierto para casi todos los comandos del Basic

del Spectrum. RUN, LIST, CONTINUE y NEW no suelen ser de mucha utilidad en un programa, pero se pueden utilizar.

Ejercicios

1.- Ponga una sentencia LIST de modo que , cuando lo ejecute, efectúe automáticamente un listado del mismo.

2.- Escriba un programa para introducir precios e imprimir el impuesto adecuado (p.e. 15%). Ponga las sentencias PRINT de modo que el ordenador indique lo que va a hacer y le pregunte el precio de entrada con extravagante cortesía. Modifique el programa para que también pueda introducir un impuesto variable (para prevenir cambios futuros).

3.- Escriba un programa para imprimir el total de los números que introduzca. (Sugerencia : Tome dos variables denominadas total (inicialice ésta con valor 0) e item. Introduzca item, súmela a total, imprima ambas y vuelva a hacerlo otra vez).

4.- ¿ Qué harían CONTINUE y NEW en un programa ?¿ Puede imaginar alguna aplicación concreta ?.

CAPITULO III .- DECISIONES

Resumen : IF, STOP
=, <, >, <=, >=, <>

Todos los programas que hemos visto hasta ahora han sido bastante predecibles (han pasado a través de las instrucciones y de nuevo han vuelto al principio). Ello no es de mucha utilidad. En la práctica, el ordenador tendría que tomar decisiones y actuar en consecuencia. La instrucción utilizada tiene la forma IF (si) algo es verdadero, o falso, THEN (entonces) obrar en consecuencia.

Por ejemplo, utilice NEW para borrar el anterior programa del ordenador e introduzca por el teclado y ejecute, el siguiente programa (que evidentemente admite que jueguen dos personas) :

```
10 REM Adivine el número
20 INPUT a:CLS
30 INPUT "Adivine el numero",b
40 IF b=a THEN PRINT "correcto":STOP
50 IF b<a THEN PRINT "demasiado bajo.Pruebe de nuevo"
60 IF b>a THEN PRINT "demasiado alto.Pruebe de nuevo"
70 GO TO 30
```

Puede constatar que una sentencia IF adopta la forma :

IF condición THEN...
en donde "..." significa una secuencia de comandos, separados por dos puntos, en la forma habitual (:).La condición es algo que ha de determinarse como verdadero o falso ; si es verdadero, se ejecutan las sentencias del resto de la línea después de THEN ; en caso contrario se saltan y el programa ejecuta la siguiente instrucción.

Las condiciones más sencillas comparan dos números o cadenas. Pueden probar si dos números son iguales o si uno es mayor que el otro y pueden probar si dos cadenas son iguales o (aprox.) si una está antes que la otra en orden alfabético. Utilizan las relaciones =, <, >, <=, >=, y <> .

= significa "igual a". Aunque es el mismo símbolo que el = en un comando LET, se utiliza en un sentido bastante diferente.

< (SYMBOL SHIFT más R) significa "menor que", por lo que :

1<2 ; -2<-1 ; -3<1

son todas, relaciones verdaderas, pero

1<0 y 0<-2

son falsas.

La línea 40 del programa anterior compara a y b. Si son iguales, entonces el programa se interrumpe por el comando STOP. El informe en la parte inferior de la pantalla 9 STOP statement, 30:3 indica que la tercera sentencia o comando de la línea 30 ha sido ejecutada, es decir, STOP..

La línea 50 determina si b es menor que a y la línea 60 si b es mayor que a. Si una de estas condiciones es verdadera entonces, se imprime el comentario correspondiente y el programa sigue su camino hasta la línea 70 que ordena el retorno a la 30 y vuelta a empezar.

El comando CLS, borrar pantalla, en la línea 20 es para impedir que la otra persona vea el número que está introduciendo.

El símbolo > (SYMBOL SHIFT másT) significa "mayor que" y es lo mismo que < pero en sentido contrario. Puede recordar cúal es cúal porqué el extremo aguzado apunta siempre al número que se supone más pequeño.

El símbolo <= (SYMBOL SHIFT más Q), que no ha de teclearse < seguido de =, significa "menor o igual a", por lo que es como < con la salvedad de que es verdadero incluso si los dos números son iguales ; por consiguiente 2<=2 es verdadero, pero 2<2 es falso.

El símbolo >= (SYMBOL SHIFT más E) significa "mayor o igual a" y es análogo a >, con las salvedades antes indicadas.

El símbolo <>(SYMBOL SHIFT más W) significa "distinto a " y es opuesto al símbolo =.

Los matemáticos suelen escribir <=, >= y <> en la forma ≤, ≥ y ≠. Escriben también cosas como 2<3<4 para significar 2<3 y 3<4, pero ello no es posible en Basic.

Observación : En algunas versiones de BASIC, pero no en la del Spectrum, la sentencia IF puede tener la forma

IF condición THEN n°de línea.

Ello significa lo mismo que

IF condición THEN GO TO n°de línea (en el Spectrum).

Ejercicios.-

1.- Pruebe este programa :

```
10 PRINT "x" :STOP:PRINT "y"
```

Cuando lo ejecute, el ordenador visualizará x y se parará con el informe 9 STOP statement, 10:2. Ahora pulse :

CONTINUE

Podría esperar que se produjera un salto atrás al comando STOP (CONTINUE suele repetir la sentencia que se refiere en el informe). Sin embargo, en este caso ello no sería de mucha utilidad porqúé el ordenador volvería a detenerse sin visualizar y. Por consiguiente, todo está previsto para que después del informe 9 CONTINUE salte al comando después del comando STOP. Por ello, en nuestro ejemplo, después de CONTINUE, el ordenador imprime y y llega al final del programa.

CAPITULO IV.- ITERACION CON BUCLES

Resumen : FOR, NEXT, TO, STEP.

Suponga que desea introducir cinco números y sumarlos juntos. Una forma (que no le recomendamos a menos que sea masoquista) sería escribir :

```
10 LET total=0
20 INPUT a
30 LET total=total+a
40 INPUT a
50 LET total=total+a
60 INPUT a
70 LET total=total+a
80 INPUT a
90 LET total=total+a
100 INPUT a
110 LET total=total+a
120 PRINT total
```

Este método no es una buena práctica de programación. Puede ser tolerable para cinco números, pero puede imaginar cuan trabajoso sería el procedimiento para sumar diez números y, en el caso de tener que sumar un centenar sería imposible.

Mucho mejor es establecer una variable para contar hasta 5 y luego, interrumpir el programa como en el ejemplo siguiente (que le recomendamos probar).

```
10 LET total=0
20 LET cuenta=1
30 INPUT a
40 REM cuenta=n°de veces que a se ha introducido
   hasta ahora
50 LET total=total+a
60 LET cuenta=cuenta+1
70 IF cuenta<=5 THEN GO TO 30
80 PRINT total
```

Observe cuán fácil es cambiar la línea 70 de modo que este programa sume diez números o un centenar.

Esta clase de contaje es tan útil que hay dos comandos especiales para hacerlo más fácil : los comandos FOR y NEXT. Siempre se utilizan conjuntamente, aunque en instrucciones separadas.

Con el empleo de estos nuevos medios, el programa que acaba de introducir por el teclado realiza exactamente lo mismo que :

```
10 LET total=0
20 FOR c=1 TO 5
30 INPUT a
40 REM c= número de veces que se ha introducido a
   hasta ahora
50 LET total=total+a
60 NEXT c
80 PRINT total
```

(Para obtener este programa a partir del anterior, ha de corregir las líneas 20,40, 60 y 70. TO es SYMBOL SHIFT con F).

Observe que hemos cambiado cuenta por c. La variable de contaje (o variable de control) de un bucle FOR-NEXT debe tener una sola letra.

El efecto de este programa es que c opera a través de los valores 1 (el valor inicial), 2, 3, 4 y 5 (el límite) y para cada uno, se ejecutan

las líneas 30, 40 y 50. A continuación, cuando c ha terminado con sus cinco valores, se ejecuta la línea 80.

Una sutileza adicional es que la variable de control no tiene que incrementarse forzosamente de uno en uno sino que puede cambiar este 1 a cualquier otro valor que desee mediante el uso de STEP en el comando FOR. La forma más general para un comando FOR es :

FOR variable de control=valor inicial TO límite STEP p

en donde la variable de control es una sola letra y el valor inicial, límite y paso son , en su totalidad, elementos que puede calcular el ordenador como números (números reales), sumas o los nombres de variables numéricas previamente introducidas en el programa. Por consiguiente, si sustituye la línea 20 del programa por :

```
20 FOR c=1 TO 5 STEP 3/2
```

entonces c operará con los valores 1, 2.5 y 4. Observe que no tiene que limitarse a números enteros y también que la variable de control no tiene porque alcanzar el límite exacto (mantiene la iteración por bucle en tanto que sea inferior o igual al límite).

Pruebe el siguiente programa para imprimir los números del 1 al 10 en orden inverso.

```
10 FOR n=10 TO 1 STEP -1
20 PRINT n
30 NEXT n
```

Anteriormente dijimos que el programa mantiene la iteración por bucle mientras que la variable de control sea inferior o igual al límite. Si profundiza en lo que ello significaría en este caso, constará que no se trata de algo superficial. La regla normal ha de modificarse cuando el paso es negativo pues entonces, el programa realiza la iteración por bucle en tanto que la variable de control sea superior o igual al límite.

Debe vigilarse cuando se están realizando dos bucles FOR-NEXT en el mismo programa, uno en el interior del otro. Pruebe el siguiente programa que imprime los números correspondientes a un juego completo de dominó (de seis puntos).

```
10 FOR m=0 TO 6
20 FOR n=0 TO m
30 PRINT m;":";n;"";      } bucle n
40 NEXT n                 } bucle m
50 PRINT
60 NEXT m
```

Puede observar que el bucle n está completamente en el interior del bucle m (se dice que están adecuadamente "encajados"). Lo que debe evitarse es tener dos bucles FOR-NEXT que se solapen sin estar completamente uno en el interior del otro, como en el caso que se indica a continuación

```
 5 REM este programa no funcionará
10 FOR m=0 TO 6
20 FOR n=0 TO m
30 PRINT m;":";n;"";      } bucle m     } bucle n
40 NEXT m
50 PRINT
60 NEXT n
```

Dos bucles FOR-NEXT deben estar uno en el interior del otro o completamente separados.

Otra cosa a evitar es saltar a la parte media de un bucle FOR-NEXT desde el exterior. La variable de control sólo está adecuadamente establecida cuando ejecuta su sentencia FOR y, si la omitiera, la sentencia NEXT produciría confusión en el ordenador. Probablemente obtendrá un informe de

error mediante el mensaje NEXT without FOR (NEXT sin FOR) o variable not found (variable no hallada).

No hay nada que le impida emplear FOR y NEXT como comandos directos. Por ejemplo :

```
FOR m=0 TO 10:PRINT m:NEXT m
```

Puede utilizarse , a veces, como una forma algo ficticia de compensar la imposibilidad de usar la función GO TO en el interior de una línea (por no usarse número de línea en el interior de ésta). Por ejemplo en el caso

```
FOR m=0 TO 1 STEP 0:INPUT a:PRINT a:NEXT m
```

El STEP (paso) de 0, hace que dicho comando se repita a sí mismo indefinidamente.

Este procedimiento no es recomendable porqué , si se produce un error, entonces habrá perdido el comando y tendrá que teclearlo de nuevo (no actuando el comando CONTINUE)

EJERCICIOS .-

1.- Una variable de control no solamente tiene un nombre y un valor como una variable ordinaria sino también un límite, un paso y una referencia a la sentencia que sigue a la correspondiente sentencia de FOR. Compruebe Vd. mismo que cuando se ha ejecutado dicha sentencia FOR toda la información está disponible (empleando el valor inicial como el primer valor que toma la variable) y también que ésta información es suficiente para que la instrucción NEXT sepa en cuanto tiene que incrementar el valor, si ha de retroceder y donde.

2.- Ejecutar el tercer programa anterior y luego entre :

```
PRINT c
```

¿ Por qué la respuesta es 6 y no 5 ?

(Porqué el comando NEXT en la línea 60 se ejecuta cinco veces y cada vez se añade 1 a c. La última vez, c se hace 6 y, a continuación, el comando NEXT no realiza el bucle hacia atrás, al haber sobrepasado c su límite).

¿ Qué sucede si pone STEP 2 en la línea 20 ?

3.-Cambie el tercer programa de manera que en vez de sumar automáticamente cinco números, le pregunte cuantos números quiere sumar. Cuando ejecute este programa, que sucede si entra 0 indicando que no desea sumar ningún número ?.Porqué supone que ha de causar confusión en el ordenador estando como está claro lo que quiere indicar? (el ordenador ha de efectuar una búsqueda en el comando NEXT c cuando, habitualmente, no es necesaria). De hecho todo ello está previsto.

4.En la línea 10 del cuarto programa, cambie el 10 por 100 y ejecute el programa. Imprimirá los números del 100 al 89 en la pantalla y preguntará scroll? en la parte inferior de la pantalla. Esto es para darle la oportunidad de ver los números que están a punto de desaparecer de la pantalla por la parte superior. Si pulsa n, STOP o BREAK, el programa se detendrá con el informe D BREAK-CONT repeats. Si pulsa cualquier otra tecla, imprimirá otras 22 líneas y repetirá la pregunta anterior.

5.- Elimine la línea 10 del cuarto programa. Cuando ejecute el nuevo programa resultante, el ordenador imprimirá el primer número y se detendrá con el informe 0 OK. Si pulsa :

```
NEXT n
```

el programa recorrerá otra vez el bucle e imprimirá el siguiente número.

CAPITULO V.- SUBRUTINAS

Resumen : GO SUB, RETURN.

A veces, en diferentes partes del programa hay que efectuar operaciones similares y se puede encontrar tecleando las mismas líneas una y otra vez. No se preocupe, no es necesario. Puede teclear estas líneas una sola vez en una forma conocida como una subrutina y luego usarlas en cualquier lugar del programa sin tener que teclearlas de nuevo.

Para hacer ésto, debe usar las sentencias GO SUB (GO to SUBrutine) y RETURN.

Se emplea, para ello, la forma :

```
GO SUB n
```

donde n es el número de la primera línea de la subrutina. Es similar a GO TO n excepto en que el computador recuerda donde estaba la instrucción GO SUB y vuelve a ella una vez efectuada la subrutina. Ello se logra poniendo el número de línea y el numero de la sentencia dentro de la línea (ambos constituyen la dirección de retorno) en la parte superior del conjunto de GO SUB.

RETURN toma la dirección de retorno superior del conjunto GO SUB y la lleva a la sentencia que le sigue.

A modo de ejemplo, examinemos de nuevo el programa para adivinar números de la pagina 1Ø. Vuelva a teclear lo siguiente :

```
1Ø REM un juego de adivinar remodelado
2Ø INPUT a:CLS
3Ø INPUT "adivine el número ",b
4Ø IF a=b THEN PRINT "Correcto": STOP
5Ø IF a>b THEN GO SUB 1ØØ
6Ø IF a<b THEN GO SUB 1ØØ
7Ø GO TO 3Ø
1ØØ PRINT "Pruebe de nuevo"
11Ø RETURN
```

La sentencia GO TO de la línea 7Ø es vital porqué, en caso contrario el programa entraría en la subrutina, causando un error (7 RETURN without GO SUB) cuando llegara a la instrucción RETURN. Pruébelo.

He aquí otro sencillo programa que muestra el uso de GO SUB

```
1ØØ LET x=1Ø
11Ø GO SUB 5ØØ
12Ø PRINT s
13Ø LET x=x+4
14Ø GO SUB 5ØØ
15Ø PRINT s
16Ø LET x=x+2
17Ø GO SUB 5ØØ
18Ø PRINT s
19Ø STOP
5ØØ LET s=Ø
51Ø FOR y=1 TO x
52Ø LET s=s+y
53Ø NEXT y
54Ø RETURN
```

Cuando ejecute este programa, trate de determinar lo que está efectuando. La subrutina empieza en la línea 5ØØ.

Una subrutina puede llamar a otra o incluso a sí misma (en este último caso se llama recursiva), así que no se preocupe de ver varias superpuestas.

CAPITULO VI.-READ, DATA, RESTORE

Resumen : READ, DATA, RESTORE

En algunos programas pasados vimos que la información o datos pueden entrarse directamente en el computador mediante la instrucción INPUT. A veces ello puede hacerse particularmente pesado especialmente cuando los datos a entrar se repiten a lo largo del programa. Puede ahorrarse un montón de trabajo y tiempo empleando las instrucciones READ, DATA y RESTORE. Por ejemplo :

```
1Ø READ a,b,c
2Ø PRINT a,b,c
3Ø DATA 1Ø,2Ø,3Ø
4Ø STOP
```

Una instrucción READ consiste en esta palabra seguida por una lista de variables separadas por comas. Funciona de forma parecida a la instrucción INPUT excepto que , en lugar de entrar los valores de las variables por el teclado, el computador los busca por sí mismo en la instrucción DATA..

Cada instrucción DATA es una lista de expresiones, numéricas o de cadenas, separadas por comas. Puede ponerla en cualquier parte del programa que prefiera, ya que el computador la ignora a menos que esté ejecutando un READ. No obstante, todos los DATA del programa acostumbran a juntarse para formar una larga lista de expresiones : la lista de datos. La primera vez que el computador (READ) lee un valor, toma la primera expresión de la lista DATA; la próxima vez toma la segunda y así, a medida que va encontrando sucesivas instrucciones READ va abriéndose camino a través del DATA list (si intenta ir más allá del final de esta lista se producirá un informe de error).

Nótese que es perder el tiempo entrar instrucciones DATA como comando directo pues READ no las encontrará. Las sentencias DATA tienen que ir en el programa con su correspondiente número de línea.

Veamos como se integran en el programa que acaba de introducir con el teclado. La línea 1Ø dice al ordenador que efectúe la lectura de tres valores de DATA,asignándoles las variables a, b y c. A continuación la línea 2Ø le dice que imprima estas variables. La instrucción DATA de la línea 3Ø suministra los valores de a,b y c. La línea 4Ø detiene el programa. Para ver el orden en que se realizan las cosas sustituya la línea 2Ø por :

```
2Ø PRINT b,c,a
```

La informaciónen DATA puede formar parte de un bucle FOR...NEXT. Entre :

```
1Ø FOR n=1 TO 6
2Ø READ D
3Ø DATA 2,4,6,8,1Ø,12
4Ø PRINT D
5Ø NEXT n
6Ø STOP
```

Cuando ejecute este programa(RUN) puede ver la instrucción READ moviéndose a través dela lista de DATA. La instrucción DATA también puede contener variables de cadena (strings). Véase un ejemplo :

```
1Ø READ d$
2Ø PRINT"La fecha de hoy es",d$
3Ø DATA "11 de octubre de 1982"
4Ø STOP
```

Esta es la manera sencilla de buscar expresiones a través de la

lista de DATA : empezar por el principio y seguir a través de ella hasta llegar al final. De todos modos puede hacer que el computador salte a cualquier lugar de la lista de DATA mediante la instrucción RESTORE. Dicha instrucción la forma la palabra RESTORE seguida por un número de línea y hace que las subsiguientes instrucciones READ empiecen a leer sus datos a partir de la primera instrucción DATA de o después de la línea cuyo número lleva aparejado. Puede Vd. omitir dicho número de línea, en cuyo caso es como si hubiera entrado el número de la primera línea del programa.

Pruebe el siguiente programa :

```
10 READ a,b
20 PRINT a,b
30 RESTORE 10
40 READ x,y,z
50 PRINT x,y,z
60 DATA 1,2,3
70 STOP
```

En este programa, los datos requeridos por la línea 10 hacen a=1 y b=2. La instrucción RESTORE 10 restaura (reset) las variables y permite que sean objeto de lectura (READ) las variables x,y,z comenzando a partir del primer número de la instrucción DATA. Reejecute este programa omitiendo la línea 30 y observe lo que ocurre.

CAPITULO VII.- EXPRESIONES

Resumen : Operaciones +, -, *, /.
Expresiones, notación científica, nombres de variables.

Ya se han visto algunas de las maneras en que el ZX Spectrum puede calcular con números. Puede efectuar las cuatro operaciones básicas (recuerde que * se emplea para multiplicar y / para dividir) y puede determinar el valor de una variable, dado su nombre.

El ejemplo :

```
LET Tasas= suma * 15/100
```

proporciona apenas una leve muestra de como pueden combinarse estos cálculos. Así pues una combinación como por ejemplo suma*15/100 se denomina una expresión que es una forma abreviada de indicar al computador que efectúe determinados cálculos, uno a continuación del otro. En nuestro ejemplo, la expresión suma*15/100 significa " determina el valor de la variable llamada suma , multiplica este valor por 15 y divídelo por 100".

Si todavía no lo ha hecho, le recomendamos que eche un vistazo al manual de introducción para conocer como el ZX Spectrum trabaja con los números y el orden en que el computador evalúa las expresiones matemáticas.

Como recordatorio :

Las multiplicaciones y las divisiones se evalúan primero. Tienen prioridad sobre la suma y la resta. En su relación mutua, la multiplicación y la división tienen la misma prioridad lo cual significa que dichas operaciones se efectúan en orden de izquierda a derecha. Una vez efectuadas siguen a continuación las sumas y restas que también tienen la misma prioridad entre si y se realizan en el mismo orden de izquierda a derecha.

A pesar de que lo que realmente necesita saber es si una operación tiene o no prioridad sobre otra, el computador asigna un número de 1 al 16 para determinar las prioridades. Por ejemplo * y / tienen prioridad 8 y + y - prioridad 6.

Este orden de cálculo es inamovible pero puede eludirlo empleando paréntesis : todo lo que esté entre paréntesis se evalúa primero y tratado como un número sencillo.

Las expresiones son muy útiles porque, cuando el computador está esperando la introducción de un número, puede dársele en su lugar una expresión y el ordenador hará el cálculo correspondiente. Hay muy pocas excepciones a esta regla y se irán viendo sobre la marcha.

Puede reunir tantas cadenas (strings) o cadenas variables como quiera en una sola expresión e incluso utilizar paréntesis.

Debemos ahora establecer lo que puede y lo que no puede usarse como nombres de variables. Como hemos dicho ya, el nombre de una cadena variable (string) debe ser una sola letra seguida de $, y el nombre de la variable de control de un bucle FOR..NEXT debe ser una sola letra, pero los nombres de las variables numéricas ordinarias pueden ser mucho más libres. Pueden emplear letras o dígitos siempre que el primer carácter sea una letra. Puede colocar espacios,si quiere, para facilitar la lectura, sin considerarlos como partes del nombre. Asímismo no se establece ninguna diferencia para el nombre tanto si está en letras mayúsculas como en minúsculas.

Veamos algunos nombres de variables admitidos.:

```
x
t42
este nombre es tan largo que se cansará de escribirlo
```

ahora somos seis
aHoRASomosSEIs

(estos dos últimos nombres se consideran los mismos y se refieren a la misma variable).

Los siguientes nombres no se permiten como nombres de variables :

2ØØ1	(comienza con un dígito)
3 osos	(idem idem)
M*A*S*H	(* no es letra ni dígito)
EA3-RCB	(- no es letra ni dígito)

Las expresiones numéricas pueden representarse por un número y un exponente (revise el manual de introducción a este respecto). Entre lo siguiente para comprobarlo :

```
PRINT 2.34eØ
PRINT 2.34e1
PRINT 2.34e2
```

y sucesivamente hasta

```
PRINT 2.34e15
```

Observará que, después de unos instantes, el computador también empieza a usar la notación científica. Esto es a causa de que no pueden emplearse números mayores de catorce caracteres. Pruebe :

```
PRINT 2.34e-1
PRINT 2.34e-2
```

y así sucesivamente

PRINT da solamente ocho dígitos significativos de un número. Pruebe:

```
PRINT 4294967295,4294967295-429e7
```

Esto prueba que el computador puede trabajar con los dígitos de 4294967295, incluso aunque no pueda visualizarlos todos a la vez.

El ZX Spectrum emplea coma flotante, lo que significa que puede mantener separados los dígitos de un número (mantisa) y la posición del punto (exponente). Ello no es siempre exacto, incluso para números enteros. Entre :

```
PRINT 1e1Ø+1-1e1Ø,1e1Ø-1e1Ø+1
```

Los números trabajan con una precisión de 9 dígitos y medio por lo que 1e1Ø es demasiado grande para ser almacenado con una precisión total. La inexactitud (cercana a 2) es superior a 1 y por ello los números 1e1Ø y 1e1Ø+1 son, para el computador, aparentemente iguales.

Para un ejemplo todavía más curioso, entre :

```
PRINT 5e9+1-5e9
```

Aquí, la inexactitud en 5e9 es cercana a 1 y, al añadir otro 1 el resultado se redondea a 2. Los números 5e9+1 y 5e9+2 son, para el computador, aparentemente iguales.

El mayor número entero que puede obtenerse con completa exactitud es el inmediatamente anterior a 2↑32 (2 elevado a 32 menos 1) es decir, 4.294.967.295.

La cadena (string) "" sin ningún carácter en absoluto se llama cadena nula o vacía. Recuerde que los espacios son significativos y no es lo mismo una cadena vacía("") que otra que contenga sólo espacios(" ")

Pruebe :

```
PRINT " Has terminado"El Quijote" por fin ?"
```


Cuando pulse ENTER obtendrá el interrogante intermitente que indica error en alguna parte de la línea. Cuando el computador encuentra las comillas del principio de "El Quijote", cree que éstas indican el final de la cadena "Has terminado" y no sabe que hacer ni entiende" El Quijote".

Hay una notación especial para solucionar esto. Siempre que quiera escribir unas comillas en el interior de una cadena, debe duplicarlas, de la siguiente manera :

```
PRINT " Has terminado ""El Quijote"" por fin ?"
```

Como puede ver las comillas son las mismas pero tecleadas dos veces en lugar de una para que el computador las reconozca.

CAPITULO VIII.- CADENAS(STRINGS)

Resumen :

Fragmentación, usando TO. Esta notación no es BASIC standart.

Dada una cadena, podemos formar una subcadena de la misma tomando secuencialmente algunos caracteres consecutivos de la misma. Por consiguiente "cadena" es una subcadena de "cadena mayor" pero no lo son ni "cad mayor" ni "cadena yor".

Hay una notación llamada "slicing" (fragmentación, partición) para describir las subcadenas y puede aplicarse arbitrariamente a cualquier tipo de cadena. La forma general es :

expresión de cadena (comienzo TO final)

así pues, por ejemplo :

```
"abcdef"(2 TO 5) = "bcde"
```

Si omite el principio, se supone 1. Si omite el final se asume el total de la cadena. Entonces :

```
"abcdef"(TO 5) ="abcdef"(1 TO 5)= "abcde"
"abcdef"(2 TO) ="abcdef"(2 TO 6)= "bcdef"
"abcdef"( TO ) ="abcdef"(1 TO 6)= "abcdef"
```

La última expresión también puede expresarse como "abcdef"().
Otra forma de expresión omite TO e inserta solamente un número.

```
"abcdef"(3) = "abcdef"(3 TO 3) = "c"
```

Aunque, usualmente el principio y el final se refieren a partes existentes de la cadena, esta regla es derogada por otra : si el principio es mayor que el final, entonces el resultado es una cadena vacía. Así:

```
"abcdef"(5 TO 7)
```

da error 3 Subscript wrong porque la cadena solo contiene 6 caracteres y 7 es demasiado grande, pero

```
"abcdef"(8 TO 7)= ""
```

y

```
"abcdef"(1 TO 0)= ""
```

El principio y el final no pueden ser negativos o obtendrá el error B integer out of range (B número entero fuera de rango). El siguiente programa ilustra algunas de estas reglas.

```
10 LET a$="abcdef"
20 FOR n=1 TO 6
30 PRINT a$(n TO 6)
40 NEXT n
50 STOP
```


Entre NEW cuando haya ejecutado este programa y pruebe este otro.

```
10 LET a$="Ya lo vere"
20 FOR n=1 TO 10
30 PRINT a$(n TO 10),a$((10-n) TO 10)
40 NEXT n
50 STOP
```

Para variables de cadena, no solamente podemos extraer subcadenas sino incluso asignarlas entre sí.
Por ejemplo, entre :

```
LET a$="Yo soy el ZX Spectrum"
```

luego

```
LET a$(5 TO 8)="******"
```

y

```
PRINT a$
```

Observe que, dado que la subcadena a$(5 TO 8) tiene sólo 4 caracteres de longitud, solamente se han empleado las primeras cuatro estrellas. Esta es una característica de la asignación a subcadenas : la subcadena ha de tener exactamente la misma longitud después que antes de su asignación. Para asegurarse de que esto realmente ocurre así la cadena que va a ser asignada debe cortarse por la derecha si es demasiado larga o rellenada con espacios si es demasiado corta. Este es el llamado método de Procrusto en memoria del posadero así llamado, que estiraba a sus clientes en el potro cuando eran más cortos que la cama o les cortaba los pies cuando eran más largos.
Si ahora prueba

```
LET a$=" Hola que tal?"
```

y

```
PRINT a$;","
```

observará que vuelve a repetirse lo mismo (esta vez con espacios) ya que a$() cuenta como una subcadena.
Las expresiones de cadenas complicadas necesitarán paréntesis antes de proceder a su fragmentación. Por ejemplo :

```
"abc"+"def"(1 TO 2) ="abcde"
("abc"+"def")(1 TO 2)="ab"
```

Ejercicio.-

1.- Pruebe a escribir un programa para imprimir el día de la semana usando fragmentación de cadenas. Sugerencia :Hacer la cadena de la forma LunMarMieJueVieSabDom.

CAPITULO IX .- FUNCIONES

Resumen : DEF
LEN, STR$, VAL, SGN, ABS, INT, SQR.
FN

Imagínese una máquina de picar carne. Si pone un trozo de carne en un extremo, girando una manivela obtendrá pulpa de carne en el otro extremo. Cada tipo de carne que ponga le proporcionará una pulpa del tipo de carne introducido.

Las funciones operan de manera algo parecida a dichas máquinas pero con una ligera diferencia : trabajan con números y cadenas en lugar de carne. Usted introduce un valor (llamado argumento), efectúe con él algunas operaciones y, eventualmente, obtendrá un valor distinto (llamado resultado).

Entrada de carne ⟶ Máquina de picar ⟶ Prod. elaborado

Entrada argumento ⟶ FUNCION ⟶ Salida resultado

Argumentos diferentes dan resultados diferentes y, si el argumento es totalmente inapropiado, la función se interrumpirá proporcionando un informe de error.

Al igual que existen diferentes máquinas para elaborar diferentes productos, también existen diferentes funciones para distintos cálculos. Cada una tiene su propio valor para distinguirlas de las demás.

Puede usar una función en expresiones simplemente tecleando su nombre seguido de su argumento y, cuando la expresión es evaluada, se obtiene el resultado de la función.

Como ejemplo, tenemos la función llamada LEN que proporciona la longitud de la cadena a la que la apliquemos. Su argumento es la cadena cuya longitud queremos hallar y el resultado es dicha longitud, así que si escribe

```
PRINT LEN "Sinclair"
```

el computador responderá 8 que es el número de letras existentes en la palabra "Sinclair" (Para obtener LEN debe pulsar al mismo tiempo CAPS SHIFT y SYMBOL SHIFT para cambiar el cursor de L a E y luego pulsar la tecla L).

Si mezcla funciones y operacionesen una sola expresión, las funciones tendrán prioridad sobre las operaciones. De nuevo, en este caso, puede obviar esta regla utilizando paréntesis. Por ejemplo, aquí tenemos dos casos que difieren solamente en los paréntesis y por tanto, los cálculos han sido efectuados enteramente diferentes en cada caso (aunque, y como por ensalmo, los resultados finales coinciden).

```
LEN "Juan"+LEN "Sierra"          LEN ("Juan"+"Sierra")
  4 + LEN "Sierra"               LEN ("JuanSierra")
  4 + 6                          LEN "JuanSierra"
    10                               10
```

Veamos, ahora, otras funciones.

STR$ convierte números en cadenas : su argumento es un número y su resultado es la cadena que aparecería en la pantalla si el número se visualizara mediante una sentencia PRINT. Nótese que su nombre termina en $ para indicar que es el resultado de una cadena. Por ejemplo, se podría decir :


```
LET a$=STR$ 1e2
```

que tendría exactamente el mismo significado que

```
LET a$="100"
```

o también podría decir

```
PRINT LEN STR$ 100.0000
```

y obtener la respuesta 3 , dado que STR$ 100.0000="100"

VAL es igual que STR$ pero en inverso: convierte cadenas en números.Por ejemplo ,

```
VAL "3.5"=3.5
```

En cierto sentido VAL es el inverso de STR$ porqué, si toma un número cualquiera, le aplica STR$ y luego le aplica VAL a su resultado, vuelve a obtener el número original.

Si toma una cadena, aplica VAL y STR$ a su resultado no siempre obtendrá la cadena original.

VAL es una función extremadamente potente ya que la cadena que es su argumento no se ve restringida a ser un número simple sino que puede ser cualquier expresión numérica. Así, por ejemplo

```
VAL "2*3"=6
```

o incluso

```
VAL ("2"+"*3")=6
```

Hay dos procesos en acción en este caso. En el primero, el argumento de VAL es evaluado como una cadena : la expresión de cadena "2"+"*3" es evaluada para dar la cadena "2*3". Luego la cadena ve eliminadas las dobles comillas y lo que está a la izquierda se evalúa como un número : así que 2*3 se interpreta para dar el número 6..

Esto puede ser ligeramente inquietante si no tomamos la cosa con filosofía y presencia de ánimo ; por ejemplo :

```
PRINT VAL"VAL""VAL""""2""""""""
```

(Recuerde que dentro de una cadena las comillas deben escribirse por duplicado. Si profundiza en las cadenas encontrará que incluso debe llegar a cuadriplicarlas e incluso octuplicarlas).

Existe otra función, bastante parecida a VAL, aunque algo menos útil, llamada VAL$. Su argumento es todavía una cadena y su resultado también. Para ver como trabaja, recuerde que VAL funciona en dos etapas : primero su argumento es evaluado como una cadena, luego las comillas desaparecen y lo que queda a la izquierda es evaluado como un número. Con VAL$, el primer paso es el mismo, pero después que las comillas han desaparecido en la segunda etapa, lo que quede en la izquierda sigue siendo tratado como una cadena. Así

```
VAL$""""Naranjada"""" = "Naranjada"
```

Observe que las comillas proliferan de nuevo. Entre

```
LET a$="99"
```

e imprima todo lo siguiente : VAL a$, VAL"a$",VAL""""a$"""",VAL a$,VAL$"a$" y VAL$""""a$"""". Algunas de estas expresiones son válidas, otras no ;intente explicar los resultados. (Mantenga la cabeza fría).

SGN es la función signo. Es la primera función que vemos que no tiene nada que ver con las cadenas, a causa de que su argumento y su re-

sultado son números. El resultado es +1 si el argumento es positivo, 0 si el argumento es 0 y -1 si el argumento es negativo.

ABS es otra función cuyo argumento y resultado son números. Convierte el argumento en un número positivo (que es el resultado) prescindiendo del signo, de modo que, por ejemplo :

ABS -3.2=ABS 3.2= 3.2

INT significa "parte entera" (un entero es un número sin decimales y puede ser incluso negativo). Esta función convierte un número fraccionario en otro entero, suprimiendo la parte fraccionaria, por ejemplo :

INT 3.9 = 3

Tenga cuidado cuando trabaje con números negativos porqué siempre redondea por abajo : así, por ejemplo,

INT -3.9= -4

SQR calcula la raiz cuadrada de un número, cuyo resultado multiplicado por si mísmo da el argumento. Por ejemplo :

SQR 4= 2 porque 2x2=4
SQR 0.25 =0.5 porque 0.5x0.5=0.25
SQR 2=1.4142136 (aprox.) porque 1.4142136x1.4142136

Si multiplica cualquier número (incluso negativo) por si mismo, la respuesta es siempre positiva. Esto quiere decir que los números negativos no tienen raiz cuadrada ; por tanto si aplica SQR a un argumento negativo obtendrá un informe de error An Invalid Argument (argumento no válido).

También puede definir funciones por su cuenta. Nombres válidos para ello son FN seguido por una letra (si el resultado es un número) o FN seguido de una letra y $ (si el resultado es una cadena). El empleo de los paréntesis es más estricto : el argumento debe estar encerrado entre paréntesis.

Puede asimismo definir una función colocando una sentencia DEF en alguna parte del programa. Por ejemplo aqui tenemos el empleo de una función FN s cuyo resultado es el cuadrado del argumento

10 DEF FN s(x)=x*x :REM el cuadrado de x

DEF se obtiene en modo extendido, usando SYMBOL SHIFT y 1. Cuando pulse esto, el computador le dará FN automáticamente, ya que en una sentencia DEF, las siglas FN siempre van incluidas a continuación. Después de esto, la s completa el nombre FN s de la función.

La x entre paréntesis es un nombre que se refiere al argumento de la función. Puede emplear cualquier letra que desee o, si el argumento es una cadena, cualquier letra seguida de $.

Después del signo = sigue la definición real de la función. Esta puede ser cualquier expresión y puede referirse también al argumento usando el nombre asignado (en este caso, x) al igual que si fuera una variable ordinaria.

Una vez entrada esta línea, puede usar esta función igual que si fuera una propia del computador, tecleando su nombre, FN s, seguido por el argumento. Recuerde que cuando haya definido su propia función, el argumento debe ir encerrado entre paréntesis. Pruébelo unas cuantas veces

PRINT FN s(2)

PRINT FN s(3+4)
PRINT 1 + INT FN s(LEN "pollo"/2+3)

Una vez haya puesto la correspondiente sentencia DEF en el programa, puede usar sus propias funciones en expresiones con la misma libertad que las propias del ordenador.

Nota.- En algunas versiones de BASIC debe además encerrar entre paréntesis el argumentó de una de las funciones del computador. En el BASIC del ZX Spectrum no sucede así.

INT siempre redondea por abajo. Para redondear al entero más cercano, añadir primero 0.5 (una función para hacer esto sería la siguiente :

20 DEF FN r(x)=INT(x+0.5) : REM redondea x al entero más cercano

entonces obtendrá, por ejemplo :

FN r(2.9)=3 FN r(2.4)=2
FN r(-2.9)=-3 FN r(-2.4)=-2

Compare estos resultados con los que obtiene empleando INT en lugar de FN r. Entre y ejecute lo siguiente :

```
10 LET x=0:LET y=0: LET a=10
20 DEF FN p(x,y)=a+x*y
30 DEF FN q()=a+x*y
40 PRINT FN p(2,3),FN q()
```

Hay un montón de puntos sutiles en este programa.

Ante todo, una función no se ve restringida a un solo argumento : puede tener varios e, incluso, ninguno, aunque deben guardarse los paréntesis. En segundo lugar, no tiene la menor importancia el punto del programa donde se introducen las sentencias DEF. Después de que el computador ha ejecutado la línea 10, simplemente salta sobre la 20 y la 30 para ejecutar la 40. No obstante, tienen que estar en una línea de programa y no pueden entrarse por comando directo.

En tercer lugar, x e y son ambas nombres de variables del programa y, al mismo tiempo, nombres de argumentos para la función FN p. Esta función "olvida" temporalmente las variables llamadas x e y, pero dado que no tiene ningún argumento llamado a, recuerda a su vez el valor de esta variable. Entonces, cuando FN p(2,3) está siendo evaluado, a tiene el valor 10 porqué es la variable, x tiene el valor 2 porqué es el primer argumento , y tiene el valor 3 por ser el segundo argumento. El resultado es, entonces, 10+2*3=16. Cuando FN q() está siendo evaluado, no tiene argumentos así que a, x e y se refieren todavía a las variables y tienen los valores 10, 0 y 0 respectivamente. La respuesta, en este caso, será 10+0*0=10

Ahora, cambie la línea 20, así :

20 DEF FN p(x,y)=FN q()

Esta vez, FN p(2,3) tendía el valor 10 porqué FN q() volverá todavía a las variables x e y en lugar de utilizar los argumentos de FN p.

Algunas versiones de BASIC (no el del Spectrum) tienen funciones llamadas LEFT$, RIGHT$, MID$ y TL$.

LEFT$ (a$,n) da la subcadena de a$ formada por los primeros n carac-

teres.

RIGHT$ (a$,n) da la subcadena de a$ formada desde n hasta el final

MID$ (a$,n1,n2) da la subcadena formada por los caracteres desde n1 hasta n2(Rectificación : formada por n2 caracteres a partir de n1)

TL$ (a$) da la subcadena de a$ formada por todos sus caracteres a excepción del primero.

Puede escribir algunas funciones definidas para efectuar el mismo trabajo : p.e.

```
10 DEF FN t$(a$) = a$(2 TO ):REM TL$
20 DEF FN I$(a$,n) = a$( TO n):REM LEFT$
```

Compruebe que esto funciona incluso con cadenas de longitud 1 y 0.

Nota.- Observe que FN I$ posee dos argumentos, un número y una cadena. Una función puede tener hasta 26 argumentos numéricos (porqué 26 ?) y, al mismo tiempo, hasta 26 argumentos-cadena.

Ejercicio .-

Emplear la funcion FN s(x) para comprobar SQR :debe encontrar que

FN s(SQR x)=x

si sustituye cualquier número positivo por x, y

SQR FN s(x)=ABS x

si x es positivo o negativo (porqué ABS ?).

CAPITULO X.- FUNCIONES MATEMATICAS

Resumen : ↑

PI, EXP, LN, SIN, COS, TAN, ASN, ACS, ATN.

Este capítulo trata de las funciones matemáticas que dispone el IX Spectrum. Muy posiblemente quizás nunca tenga que emplearlas en sus programas, así que, si encuentra las explicaciones algo farragosas, no vacile en saltarselas. Estas cubren la operación ↑ (elevar a una potencia), las funciones EXP y LN y las funciones trigonométricas SIN, COS, TAN y sus inversas, ASN, ACS, y ATN.

↑ y EXP.

Puede elevar un número a la potencia de otro (que significa multiplicar el primer número por sí mismo tantas veces como indique el segundo número). Esto normalmente se expresa escribiendo el segundo número a la derecha y arriba del primer número. Como, evidentemente ello le resulta difícil al computador se adopta unánimemente el símbolo ↑ . Por ejemplo, las potencias de 2 son :

2↑1=2
2↑2=4 (2 al cuadrado, normalmente indicado 2^2)
2↑3=2*2*2=8 (2 al cubo, normalmente indicado 2^3)
2↑4=2*2*2*2=16(2 a la cuarta potencia, normalmente expresado 2^4)

De modo elemental 'a↑b' significa "a multiplicado por sí mismo b veces", pero, obviamente, esto sólo tiene sentido si b es un número entero positivo. Para hallar una definición válida para otros valores de b, consideremos la regla :

a↑(b+c)=a↑b*a↑c

Adviértase que damos a↑ una prioridad más alta que a * y / así que, cuando haya varias operaciones en una expresión, la↑ será evaluada antes que * y /. No es difícil de aceptar que esto es así cuando b y c son ambos positivos y enteros; no obstante, si decidimos que esta regla se cúmpla aún en el caso de que no lo sean, debemos aceptar lo siguiente :

a↑0=1

a↑(-b)=1/a↑b

a↑(1/b)= la raiz b de a, es decir, el número de veces que hay que multiplicar b por sí mismo para obtener a

y, finalmente

a↑(b*c)=(a↑b)↑c

Si no conocía estas expresiones con anterioridad, no intente recordarlas de memoria ; simplemente recuerde que :

a↑(-1) = 1/a y que,
a↑(1/2)=SQR a

y quizás con el manejo frecuente de ello las anteriores acaben por ser familiares.

Experimente con todo ello, entrando y ejecutando este programa

```
10 INPUT a,b,c
20 PRINT a↑(b+c) = a↑b*a↑c
30 GO TO 10
```

Por supuesto, si las reglas dadas anteriormente eran ciertas, cada vez los dos números que imprima el computador deben ser iguales (a causa de la forma en que trabaja el computador con ↑, el número de la izquierda, a er este caso, no debe ser nunca negativo).

Un clásico ejemplo de emplec de esta función está en el cálculo del interés compuesto. Supongamcs que invertimos una cantidad de dinero en una sociedad a un interés del 15% anual. Al cabo de un año tendrá, además del 100% que invirtió, 15% que la sociedad le ha reportado haciendo un total de 115%. Para indicarlo con otras palabras, Vd. ha incrementado su cantidad inicial multiplicándola por 1.15 Al cabo del segundo año, sucederá lo mismo así que Vd. tendrá 1.15*1.15=1.15↑2 = =1.3225 veces la cantidad inicial. En general, después de y años, tendrá 1.15↑y veces la cantidad de dinero que invirtió.

Si prueba ahora este comando

```
FOR y=0 TO 100 PRINT n,100*1.15↑y:NEXT y
```

comprobará que partiendo de unas pocas pesetas, el total aumenta rápidamente y tanto más cuanto más tiempo transcurre. (Sin embargo, no se esfuerce, la inflación es más rápida aún).

Esta manera de comportarse, donde después de un intervalo fijo de tiempo, cierta cantidad se multiplica a sí misma en una proporción fija, es llamada de "crecimiento o incremento exponencial" y se calcula elevando un número fijo a la potencia de el tiempo

Suponga que hizo lo siguiente :

```
10 DEF FN a(x) = a↑x
```

En este caso, a es más o menos fija, mediante sentencias LET su valor corresponde al tipo de interés que varía sólo muy de tiempo en tiempo.

Existe un cierto valor de a que hace a la función FN especialmente atractiva a los expertos ojos de un matemático y este valor es el llamado e. El ZX Spectrum tiene una función llamada EXP definida por

```
EXP x = e↑x
```

Desgraciadamente e, en sí mismo, no es un número particularmente atractivo : se trata de un decimal no recurrente. Puede ver unos pocos decimales del mismo entrando

```
PRINT EXP 1
```

ya que EXP 1 = e↑1= e. Por supuesto, se trata solamente de una aproximación. Nunca conseguirá escribir e con exactitud.

LN .- La inversa de una función exponencial es una función logarítmica : el logaritmo (en base a) de un número x es la potencia a la que tiene que elevar a para obtener el número x y se escribe $\log_a x$. Entonces, por definición $a \uparrow \log_a x = x$; y también es cierto que $\log(a \uparrow x) = x$.

Puede ser que ya conozca de antemano como emplear logaritmos de base 10 para efectuar multiplicaciones ; reciben el nombre de logaritmos

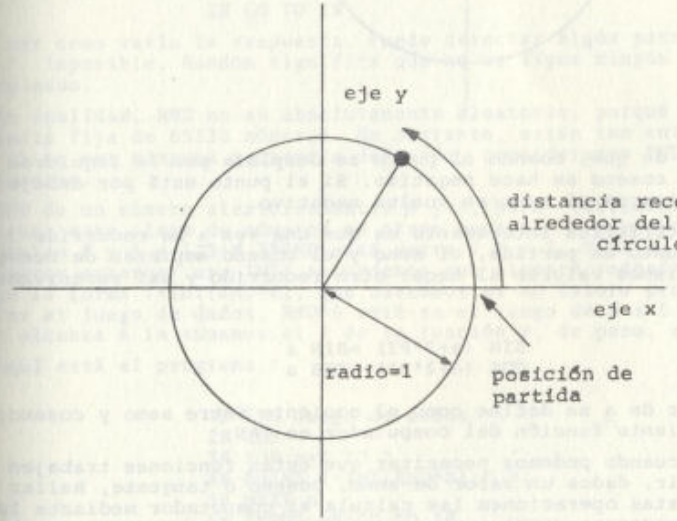
comunes. El Zx Spectrum tiene una función LN la cual calcula logaritmos en base e, llamados logaritmos naturales. Para el cálculo de logaritmos de cualquier otra base, debe dividir el logaritmo natural por el idem de la base :

$$\log_a x = LNx/LNa$$

PI.- Dado cualquier círculo, puede encontrar su perímetro (es decir, la longitud de la circunferencia que lo rodea), multiplicando su diámetro por un número llamado π .(π es la letra p del alfabeto griego, inicial de perímetro. Su nombre es PI).

Al igual que e, PI es un número racional aperiódico con infinitos decimales ; su valor es 3.141592653589... etc..- La palabra PI en el Spectrum (modo extendido + tecla M), recibe el valor de este número. Pruebe PRINT PI.

SIN, COS y TAN ;ASN, ACS y ATN.- Las funciones trigonométricas miden lo que sucede cuando un punto se mueve alrededor del círculo. Imaginemos un círculo de radio 1 (1 qué ?;no importa puesto que lo mantendremos igual en todos los cálculos) y un punto moviéndose a su alrededor. El punto empieza en la posición de las tres en el reloj y se moverá en sentido contrario a las agujas.;

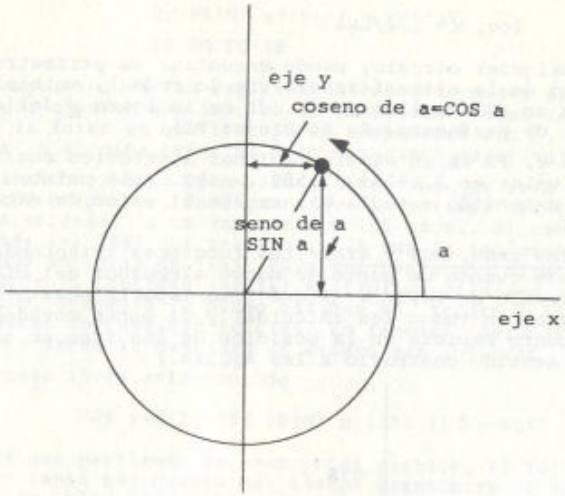


Dibujemos también dos líneas llamadas ejes en el centro del círculo. La que pasa a través de las 9 y las 3 del reloj se llama eje-x y la que pasa por las 12 y las 6, eje-y.

Para especificar la situación del punto, diremos cuánto se ha movido alrededor del círculo a partir de la posición inicial (las 3). Llamaremos a a esta distancia. Conocemos que la circunferencia del círculo es 2·π (ya que su radio es 1 y, por consiguiente, su diámetro vale 2); así pues, cuando se mueva la cuarta parte de la longitud de la circunferencia, a será igual a π/2. Si se ha movido la mitad de la circunferencia a=π y, si ha recorrido la totalidad, a= 2π .

Dada la distancia curva alrededor del perímetro, a, interesa conocer otras dos distancias que son la separación del punto respecto al eje x y respecto al eje y. Estas dos distancias mencionadas se llaman, respectivamente, Seno y Coseno de a y las calcula el computador

mediante las funciones SIN y COS.



Adviértase de que, cuando el punto se desplaza por la izquierda del eje-y, el coseno se hace negativo. Si el punto está por debajo del eje-x, entonces el seno se vuelve negativo.

Otra característica interesante es que una vez a ha recorrido 2 y vuelve al punto de partida, el seno y el coseno empiezan de nuevo, tomando los mismos valores al hacer otro recorrido y así sucesivamente.

```
SIN (a+2*PI) =SIN a
COS (a+2*PI) =COS a
```

La tangente de a se define como el cociente entre seno y coseno; la correspondiente función del computador es TAN.

De vez en cuando podemos necesitar que estas funciones trabajen al revés, es decir, dados un valor de seno, coseno o tangente, hallar el valor de a. Estas operaciones las calcula el computador mediante las funciones de arcoseno (ASN), arcocoseno (ACS) y arcotangente (ATN).

En el diagrama que muestra al punto moviéndose alrededor del círculo, obsérvese el radio que une el centro con dicho punto. Podrá ver que la distancia que hemos llamado a, la distancia que recorre el punto alrededor del círculo, es una manera de medir el ángulo que se ha desplazado el radio a partir del eje-x. Cuando a= π/2, el ángulo vale 9Ø°; cuando a= π, el ángulo es de 18Ø° y así sucesivamente hasta el valor a= 2π en que el ángulo mide 36Ø°. Como puede ver, podemos olvidar los grados y medir los ángulos en radianes. Luego tendremos que π/2 radianes = 9Ø° y así sucesivamente .

Debe recordar siempre que,en el ZX Spectrum, las funciones SIN, COS, etc..., emplean radianes y no grados. Para convertir grados en radianes, simplemente divida por 18Ø y multiplique por π y, para pasar de radianes a grados, divida por π y multiplique por 18Ø.

CAPITULO XI.- NUMEROS ALEATORIOS

Resumen : RANDOMIZE
RND

Este capítulo trata de la función RND y la instrucción RANDOMIZE. Ambas se emplean en combinación con números aleatorios, por lo que debe procurar no mezclarlas. Ambas están en la tecla T ; RANDOMIZE se ha abreviado a RAND.

En ciertos casos RND actúa como una función; hace cálculos y presenta un resultado. Sólo difiere en que no necesita un argumento.

Cada vez que utilice RND obtendrá un nuevo número aleatorio entre Ø y 1 (a veces puede ser Ø pero nunca 1).

Pruebe :

```
1Ø PRINT RND
2Ø GO TO 1Ø
```

para ver como varía la respuesta. Puede detectar algún patrón determinado ?. Imposible. Random significa que no se sigue ningún esquema predeterminado.

En realidad, RND no es absolutamente aleatorio, porqué sigue una secuencia fija de 65536 números. No obstante, están tan entremezclados que, al no ver ninguna secuencia definida, consideramos RND como pseudoaleatorio.

RND da un número aleatorio entre Ø y 1, pero fácilmente pueden conseguirse esta clase de números en otras escalas. Por ejemplo, 5*RND está entre Ø y 5 y 1.3+Ø.7*RND está entre 1.3 y 2. Para obtener solamente números enteros, use INT(recordemos que siempre redondea por defecto) de la forma 1+INT(RND*6), que usaremos es un futuro programa para simular el juego de dados. RND*6 está en el rango de Ø a 6 , pero como nunca alcanza 6 le sumamos el 1 de la función y, de paso, evitamos el Ø.

Aquí está el programa :

```
1Ø REM Juego de dados
2Ø CLS
3Ø FOR n=1 TO 2
4Ø PRINT 1+INT(RND*6);" ";
5Ø NEXT n
6Ø INPUT a$:GO TO 2Ø
```

Pulse ENTER cada vez que quiera lanzar los dados.

La sentencia RANDOMIZE se emplea para empezar aleatoriamente un programa basado en secuencias de números, como puede comprobar en el programa, comenzando siempre en el mismo lugar.

```
1Ø RANDOMIZE 1
2Ø FOR n=1 TO 5:PRINT RND,:NEXT n
3Ø PRINT:GO TO 1Ø
```

Después de cada ejecución de RANDOMIZE 1, la RND secuencia empieza siempre con Ø.ØØ22735596. Puede emplear otros números entre 1 y 65535 en la sentencia RANDOMIZE para comprobar que esto se repite en todos los casos.

Si tiene un programa con RND incluído y también algunos errores que no ha localizado, le será de utilidad el empleo de RANDOMIZE y de esta manera el programa siempre se comportará de la misma forma.

RANDOMIZE por si mismo (RANDOMIZE Ø tiene el mismo efecto), es distinta porqué realmente lo que hace es aleatorizar RND como puede ver en el siguiente programa :

```
1Ø RANDOMIZE
2Ø PRINT RND :GO TO 1Ø
```

La secuencia que obtendrá no es realmente aleatoria, ya que RANDOMIZE depende del tiempo que lleva funcionando el computador. Dado que el tiempo no varía practicamente cada vez que se ejecuta RANDOMIZE, el próximo RND será más o menos el mismo. Obtendrá mayor efecto aleatorio entrando GO TO 2Ø en lugar de GO TO 1Ø.

La mayoria de las versiones de BASIC utilizan RND y RANDOMIZE aunque no siempre con los mismos resultados.

He aqui un programa de lanzamiento de monedas contando las caras y las cruces.

```
1Ø LET caras=Ø:LET cruces=Ø
2Ø LET moneda=INT(RND*2)
3Ø IF moneda =Ø THEN LET caras=caras + 1
4Ø IF moneda =1 THEN LET cruces=cruces+1
5Ø PRINT caras;",";cruces,
6Ø IF cruces<>Ø THEN PRINT caras/cruces ;
7Ø PRINT :GO TO 2Ø
```

La relación entre caras y cruces debe ser aproximadamente de 1 si lo practica el tiempo suficiente, pues las probabilidades están al 5Ø%.

Ejercicios.-

1.- Comprobar la siguiente regla :
Suponga que escoge un número entre 1 y 872 y entre

RANDOMIZE este número

Entonces, el siguiente valor de RND será :

(75*(este número +1)-1)/65536

2.- Para expertos matemáticos(solamente)

Sea b un número primo (grande) y sea a una raiz primaria de módulo b. Entonces si c_i es el residuo de a^1 módulo b ($1 \leq c_i \leq b-1$), la secuencia

$$\frac{c_i - 1}{b-1}$$

es una secuencia cíclica de b-1 números distintos en la escala 0 a 1 (excluyendo 1). Escogiendo a adecuadamente, estos números pueden aparecer razonablemente aleatorios.

El número 65537 es un primo de FERMAT, $2^{16}+1$. A causa de que 10^5 grupos multiplicativos de residuo≠Ø, módulo 65537 tienen potencia de 2 como su orden, el residuo es una raiz primaria sólo en el caso de que no sea un residuo cuadrático. Emplee la ley de Gauss de reciprocidad

cuadrática para comprobar que 75 es una raiz primaria módulo 65537.

El ZX Spectrum utiliza b=65537 y a=75 y almacena algunos c_i-1 en la memoria. RND obliga a sustituir c_i-1 de la memoria por c_{i+1}-1, proporcionando el resultado (c_{i+1}-1),(b-1). RANDOMIZE n (con $1 \leq n \leq 65535$) hace c_i igual a n+1.

RND está bastante regularmente repartido en la escala de Ø a 1.

CAPITULO XII.- MATRICES

Resumen : MATRICES (la forma en que el ZX Spectrum trata las matrices de cadenas es ligeramente peculiar).
DIM...

Supongamos que tiene una lista de números, p.e. las puntuaciones de 10 estudiantes de una clase. Para guardarlos en el computador debería asignar una variable simple para cada persona, pero encontraría este método algo trabajoso. Llamando a las variables Bloggs 1, Bloggs 2 y así hasta Bloggs 1Ø, pero el programa para asignarles estos diez números sería bastante largo y cansado de entrar.

Mucho mas fácil sería entrar un programa de esa guisa :

```
 4 REM este programa es inviable
1Ø FOR n=1 TO 1Ø
2Ø READ Bloggs n
3Ø NEXT n
4Ø DATA 1Ø, 2 , 5, 19, 16, 3, 11, 1, Ø, 6
```

pero lamentablemente no es posible.

No obstante existe un mecanismo mediante el cual puede aplicar este principio, empleando matrices. Una matriz es un conjunto de variables, llamados sus elementos, todos con el mismo nombre y diferenciados sólo por un número (el subíndice) escrito entre paréntesis a continuación del nombre. En nuestro ejemplo, el nombre podría ser b (al igual que las variables de control en los bucles FOR..NEXT, el nombre de una matriz debe ser una letra única) y las 1Ø variables podrían ser b(1),b(2), hasta llegar a b(1Ø).

Los elementos de una matriz se llaman variables con subíndice en contraste con las variables simples que ya conocemos.

Antes de usar una matriz, debe reservar espacio para la misma en el interior del computador, empleando para ello la sentencia DIM(de dimensión).

```
DIM b(1Ø)
```

asigna a una matriz llamada b una dimensión 1Ø(p.e. hay 1Ø variables con subíndice b(1),....b(1Ø)) y asigna a los diez valores el valor 0. También anula cualquier matriz llamada b que existiera previamente.(Pero no una simple variable. Una matriz y una variable numérica con el mismo nombre pueden coexistir y no puede haber la menor confusión entre ellas ya que la matriz siempre posee un subíndice).

El subíndice puede ser una expresión numérica arbitraria, así que ahora puede escribir :

```
1Ø FOR n=1 TO 1Ø
2Ø READ b(n)
3Ø NEXT n
4Ø DATA 1Ø,2,5,19,16,3,11,1,Ø,6
```

También puede establecer matrices con más de una dimensión. En una matriz bidimensional necesita dos números para especificar cada uno de los elementos (similar a la línea y la columna para definir un punto en una pantalla) y, por tanto, tiene la forma de tabla. Alternativamente, si imagina los números de línea y columna referidos a una página impresa podría tener una dimensión adicional que sería el número de páginas. Por supuesto, estamos hablando de matrices numéricas así

que los elementos no son letras impresas de un libro sino números. Piense en los elementos de una matriz tridimensional v que queda especificada por v(nºde página, nºde línea, nºde columna).

Por ejemplo, para establecer una matriz bidimensional c con las dimensiones 3 y 6, se emplea la sentencia DIM

```
DIM c(3,6)
```

y esto le proporciona 3*6= 18 variables con subíndice.

```
c(1,1),c(1,2),...,c(1,6)
c(2,1),c(2,2),...,c(2,6)
c(3,1),c(3,2),...,c(3,6)
```

El mismo principio es válido para cualquier número de dimensiones.

A pesar de que puede tener un número y una matriz con el mismo nombre, no puede tener dos matrices con el mismo nombre, incluso aunque tengan distintas dimensiones.

Hay también matrices de cadenas. Las cadenas en una matriz difieren de las cadenas simples en que deben de tener una longitud determinada y para lograrla se aplica el principio de Procrusto, o se corta o se rellena de espacios. Otra manera de considerarlas es como matrices (con una dimensión adicional) de caracteres sencillos. El nombre de una matriz de cadenas puede ser cualquier letra seguida de $ y una matriz de cadenas no puede tener el mismo nombre que una cadena simple (al contrario que en el caso de números).

Suponga entonces que desea una matriz a$ de cinco cadenas. Debe decidir que longitud deben tener estas cadenas (supongamos 1Ø caracteres). Entre :

```
DIM a$ (5,1Ø)
```

Esto establece una matriz de 5*1Ø caracteres, pero puede considerar también cada fila como si fuera una cadena :

```
a$(1)=a$(1,1)a$(1,2)...a$(1,1Ø)
a$(2)=a$(2,1)a$(2,2)...a$(2,1Ø)
. . . . . .
. . . . . .
a$(5)=a$(5,1)a$(5,2)...a$(5,1Ø)
```

Si da el mismo número de subíndices (dos en este caso) que dimensiones en la sentencia DIM, obtendrá entonces un carácter único, pero si olvida el último fuera, tendrá una cadena de longitud fija. Así, por ejemplo, A$(2,7) es el séptimo carácter en la cadena A$(2) ; usando la notación de fragmentación (slicing) podríamos también escribirlo como A$(2)(7). Ahora entre

```
LET a$(2)="123456789Ø"
```

y

```
PRINT a$(2),a$(2,7)
```

Obtendrá :

```
123456789Ø      7
```

Para el último subíndice (el único que puede omitir) tiene también

la manera de fragmentarlo, así que, por ejemplo :

```
a$(2,4 TO 8)=a$(2)(4 TO 8)="45678"
```

Recuerde :

En una matriz de cadenas, todas las cadenas tienen la misma longitud
La sentencia DIM tiene un número adicional (el último) para especificar su longitud.
Cuando escriba una variable con subíndice para una matriz de cadena, puede colocar un número adicional o un elemento de fragmentación (slicer) para estar en correspondencia con el número adicional en la sentencia DIM

Puede tener matrices de cadena sin dimensiones. Entre

```
DIM a$(10)
```

y verá que a$ se comporta igual que una variable de cadena, excepto en su longitud que siempre será 10 con una asignación siempre "procrusteana"

Ejercicios.-

1. Use READ y DATA para establecer una matriz m$de doce cadenas en la cual m$(n) es el nombre del n-ésimo mes.(Ayuda : la sentencia DIM será DIM m$(12,9). Pruébelo imprimiendo toda la m$(n) (use un bucle).

Entre :

```
PRINT "estamos en el mes de";m$(4);" lluvias mil"
```


CAPITULO XIII.- CONDICIONES

Resumen : AND, OR
NOT

Vimos en el capítulo III como la sentencia IF tomaba la forma

```
IF condición THEN
```

Las condiciones, en aquel caso, eran las relaciones(=, < ,>=,<= y <>), que comparaban dos números o dos cadenas. Puede también combinarlas entre sí usando las operaciones lógicas AND(y), OR(o) y NOT (no).
Una relación AND (y) otra relación es verdad cuando ambas relaciones son verdad, así que puede escribir una línea como ésta :

```
IF a$="si" AND x>0 THEN PRINT x
```

en donde x solamente se imprime si a$="si" y x>0. El Basic en este caso es tan "gramatical" que no vale la pena abundar en detalles. Igual que en nuestro idioma, puede relacionar muchas expresiones con AND (y) y, el conjunto entero será verdad si todas las expresiones citadas lo son.
Una relación OR(u) otra es verdadera siempre que lo sea al menos una de las dos individualmente. Recuerde que esto es así incluso si ambas son verdad ; ello es algo que gramaticalmente no siempre se cumple.
La relación NOT invierte las cosas. La relación NOT es verdad cuando la relación es falsa y falsa cuando es verdadera.

Pueden efectuarse expresiones lógicas con las relaciones AND,OR y NOT al igual que los números pueden operar con +, -, etc.; incluso puede poner paréntesis si le conviene. Las prioridades actúan de la misma manera que en las operaciones usuales +, -,*,/ y ↑, haciendo que OR tenga la prioridad más baja, luego AND, a continuación NOT, luego las relaciones y, finalmente las operaciones usuales.

NOT es realmente una función, con un argumento y un resultado, pero su prioridad es mucho más baja que la de las otras funciones. Por tanto su argumento no necesita paréntesis a menos que contenga AND u OR (o ambas). NOT a=b significa lo mismo que NOT(a=b) (y, por supuesto, lo mismo que a<>b).

<> es la negación de = en el sentido de que es verdad si, y sólo en este caso, = sea falso. En otras palabras

```
a<>b es lo mismo que NOT a=b
```

y también

```
NOT a<>b equivale a a=b
```

Convénzase a sí mismo de que >= y <= son las negaciones de< y > respectivamente, por tanto siempre puede prescindir de NOT en una relación cambiando ésta.

También,

```
NOT(1ª expresión lógica AND 2ª expresión lógica)
```

equivale a

```
NOT (la 1ª)  OR NOT (la 2ª)
```

NOT (una primera expresión OR una segunda)

equivale a

NOT (una primera) AND NOT (una segunda)

Usando esta norma puede Vd. emplear NOT mediante paréntesis provisionalmente, hasta que llegue el momento de emplearlas y entonces prescindir de ellas por conversión. Lógicamente hablando la expresión NOT no es imprescindible aunque su empleo puede clarificar los programas.

La siguiente sección es bastante complicada y puede ser tranquilamente "saltada" si lo desea.

Pruebe :

```
PRINT 1=2,1<>2
```

y seguramente esperará un informe de error de sintáxis. De hecho, en lo que respecta al ordenador trata estas expresiones como lógicas y contesta con números ordinarios siguiendo las siguientes normas :

(i) =, <, >, <=, >= y <> dan resultados numéricos: 1 si es verdad Ø si es falso. Entonces el comando PRINT anterior dió Ø para '1=2', lo cúal es falso y 1 para '1<>2' que, evidentemente es cierto.

(ii) En

IF condición THEN...

la condición puede ser cualquier expresión numérica. Si su valor es Ø, entonces se interpreta como falsa y cualquier otro valor (no solamente 1) vale como verdadera. Luego la sentencia IF equivale exactamente a

IF condición <>0 THEN...

(iii) AND, OR y NOTson también operaciones con valoración numérica.

'x AND y'tiene el valor { x, si y es verdad (distinto de cero); Ø (falso), si y es falso (cero)

'x OR y' tiene el valor { 1 (verdad), si y es verdad (distinto a Ø); x , si y es falso (cero)

'NOT x' tiene el valor { Ø (falso), si x es verdad (distinto de cero); 1 (verdad), si x es falso (cero)

Fíjese en que'verdad' significa 'distinto de cero' cuando comprobamos un valor dado, pero equivale a '1' cuando el computador responde.

Relea de nuevo esta capítulo y asegúrese de que ha captado el fondo del mismo.

En las expresiones 'xANDy','x OR y' y 'NOT x', x e y toman normalmente los valores Ø y 1 para falso y verdad. Efectúe las diez diferentes combinaciones (cuatro para AND, cuatro para OR y dos para NOT) y compruebe que siguen las normas citadas en este capítulo.

Pruebe este programa :

```
10 INPUT a
20 INPUT b
30 PRINT (a AND a>=b)+(b AND a<b)
40 GO TO 10
```

Cada vez imprimirá el mayor de los números a y b

Trate de recordar que

x AND y

equivale exactamente a

x'si'y (si no, el resultado es cero)

y que

x OR y

significa

x 'a menos que' y (en cuyo caso el resultado es 1)

Una expresión que emplee AND u OR se llama una expresión condicional. Un ejemplo usando OR podría ser :

```
LET precio total =precio menos tasas *(1.15 OR v$= calculado cero)
```

Observe com AND tiende a ir con la suma (dado que su valor por defecto es Ø) y OR tiende a la multiplicación (ya que su valor por defecto es).

También puede valorar cadenas como expresiones condicionales, pero solamente empleando AND

'x$ AND y' equivalen a { x$ si y≠ Ø ; "" si y=Ø

lo que significa x$'si' y (en caso contrario, la cadena vacía)

Pruebe este programa, donde introduce dos cadenas y las coloca en orden alfabético :

```
10 INPUT "entrar dos cadenas" ' a$,b$
20 IF a$ >b$ THEN LET c$=a$:LET a$=b$:LET b$=c$
30 PRINT a$;" ";("<"AND a$<b$)+("="AND a$=b$);" ";b$
40 GO TO 10
```

Ejercicio.-

1.- El BASIC puede ser ligeramente distinto a su equivalente gramatical. Considere, por ejemplo, la expresión 'si a no es igual a b o c'. Como escribiría su equivalente en BASIC ?. La respuesta no es ;

```
IF A<>B OR C
```

ni, por supuesto

```
IF A<>B OR A<>C
```

CAPITULO XIV.- SET DE CARACTERES

Resumen : CODE, CHR$
POKE, PEEK
USR
BIN

Las letras, dígitos, signos de puntuación, etc. que pueden aparecer en las cadenas, se llaman caracteres y constituyen el alfabeto o 'Set de caracteres'que emplea el ZX Spectrum. La mayoría de dichos caracteres son símbolos simples, pero hay algunos llamados 'tokens' (sin equivalente adecuado en español) que representan palabras enteras, tales como PRINT, STOP, <>, etc...

Existen 256 caracteres y a cada uno le corresponde un código entre Ø y 255. Hay una lista completa de ellos en el apéndice A. Para efectuar la conversión entre código y caracteres, hay las dos funciones llamadas CODE y CHR$.

CODE se aplica a una cadena y proporciona el código del primer carácter de la misma (ó Ø si la cadena está vacía).

CHR$ se aplica a un número y proporciona el carácter cuyo código equivale a este número.

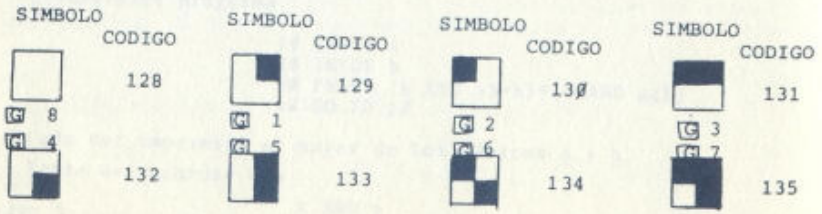
Este programa imprime el set completo de caracteres :

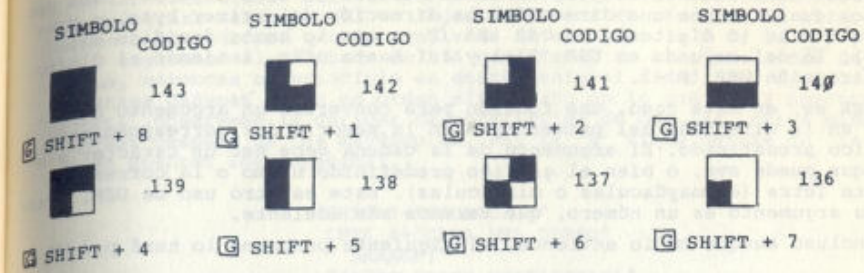
```
10 FOR a=32 TO 255:PRINT CHR$ a;:NEXT a
```

En la parte superior observamos un espacio, 15 símbolos y signos de puntuación, los diez dígitos, siete símbolos, las letras mayúsculas, 6 símbolos, las letras minúsculas y cinco símbolos más. Todos ellos están tomados (excepto £ y ©) del mundialmente empleado set de caracteres ASCII (anagrama de AMERICAN STANDARD CODES for INFORMATION INTERCHANGE); el ASCII asigna también códigos a dichos caracteres y dichos códigos son los que emplea el ZX Spectrum.

El resto de los caracteres no forman parte del ASCII y son exclusivos del Zx Spectrum. Los primeros de ellos son un espacio y 15 configuraciones de manchas negras y blancas. Estas son los llamados 'simbolos gráficos' y se emplean para formar dibujos. Puede entrarlos directamente del teclado, empleando el llamado 'graphics mode'. Si pulsa GRAPHICS (Caps Shift + 9), entonces el cursor cambia a G. Ahora las teclas de los dígitos 1 a 8 darán los símbolos graficos. Por sí mismos, proporcionan los dibujos de la tecla y, con la tecla de cambio apretada, dan el mismo gráfico pero invertido (como el positivo y negativo de una fotografía).

Prescindiendo de SHIFT, la tecla del dígito 9 le devuelve al modo normal (L) y la tecla Ø es DELETE (BORRADO).
He aquí los 16 símbolos gráficos y como se obtienen :

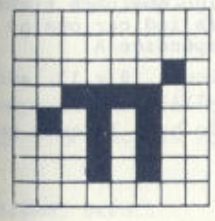




Después de los símbolos gráficos, verá que aparece una nueva copia del alfabeto de la A a la U. Estos son caracteres que puede definir por sí mismo, a su gusto, a pesar de que cuando el computador está recién conectado estos caracteres son simples letras. Se llaman gráficos definibles por el usuario. Puede obtenerlos por el teclado yendo a 'GRAPHICS MODE' como antes y luego empleando las teclas de A hasta U.

Para definir un nuevo carácter por sí mismo, siga las instrucciones y obtendrá el símbolo π (PI).

a) Decida la forma del carácter. Cada carácter está formado por una matriz de 8 x 8 puntos, cada uno de los cuales puede definirse con el color del fondo o el del texto (ver volumen de introducción). Debería dibujar un diagrama parecido al siguiente, con cuadros negros para el texto (INK):



Hemos dejado un margen de cuadros vacíos alrededor porqué las otras letras también lo tienen (excepto en las minúsculas, cuyo 'rabo' llega hasta el fondo).

b) Decida qué tecla imprimirá π cada vez que la pulse. Por ejemplo la P de forma que, cada vez que esté en modo 'GRAFICOS', pulsándola obtendrá π .

c) Almacene la nueva configuración. Cada gráfico predefinido tiene su distribución formada por ocho números, uno por cada fila. Puede escribir cada uno de dichos números como BIN seguido de ocho ceros y unos (Ø para el fondo, 1 para el texto) de manera que los ocho números para nuestro carácter serán :

```
BIN ØØØØØØØØ
BIN ØØØØØØØØ
BIN ØØØØØØ1Ø
BIN ØØ1111ØØ
BIN Ø1Ø1Ø1ØØ
BIN ØØØ1Ø1ØØ
BIN ØØØ1Ø1ØØ
BIN ØØØØØØØØ
```

(Si conoce algo acerca de números binarios, le ayudará saber que BIN se emplea para escribir un número en binario en lugar de decimal)

Estos ocho números se guardan en la memoria en ocho lugares, cada uno de los cuales tiene una dirección. La dirección del primer byte, o grupo de 8 bits (ó dígitos), es USR 'P' (P ya que lo hemos decidido así en b), la del segundo es USR 'P'+1 y así hasta ocho, teniendo el último la dirección USR 'P'+7.

USR es, en este caso, una función para convertir un argumento de cadena en la dirección del primer byte en la memoria del correspondiente gráfico predefinido. El argumento de la cadena debe ser un carácter simple que puede ser, o bien el gráfico predefinido mismo o la correspondiente letra (en mayúsculas o minúsculas). Este es otro uso de USR, cuando su argumento es un número, que veremos más adelante.

Incluso aunque no lo entienda, el siguiente programa lo hará por Vd.

```
10 FOR n=Ø TO 7
20 INPUT fila:POKE USR "p"+n,fila
30 NEXT n
```

Esperará que le introduzca ocho veces números BIN como los indicados anteriormente (u otros). Entrelos en el orden correcto, empezando por la fila superior.

La sentencia POKE almacena directamente un número en la posición de memoria, eludiendo los mecanismos empleados usualmente por el BASIC. El opuesto a POKE es PEEK y éste le permite 'ver' el contenido de una posición de memoria, a pesar de que no puede alterar dicho contenido. Ello se explicará con más detenimiento en el capítulo XXVIII.

A continuación de los gráficos predefinidos, vienen los 'tokens'.

Habrá observado que no han resultado impresos en la pantalla los primeros 32 caracteres, con código de Ø a 31. Estos son caracteres de control. No producen nada concreto en la pantalla pero tienen un determinado efecto sobre la misma o, incluso, controlan algo que nada tiene que ver con la pantalla, por tanto ésta imprime '?' para indicar que no los entiende. Están más profundamente descritos en el apéndice A.

Tres códigos que sí utiliza la televisión, son los 6, 8 y 13, aúnque en realidad, solamente CHR$ 8, le será de posible utilidad.

CHR$ 6 imprime espacios de la misma manera que lo hace la coma en una sentencia PRINT, por ejemplo :

```
PRINT 1;CHR$ 6;2
```

equivale a

```
PRINT 1,2
```

Por supuesto, no es una forma muy útil de empleo. Más sutil sería la siguiente :

```
LET a$="1"+CHR$ 6+"2"
PRINT a$
```

CHR$ 8 es 'retroceso'. Mueve la posición del cursor hacia atrás un espacio. Pruebe :

```
PRINT "1234";CHR$ 8;"5"
```

lo cual imprimirá en pantalla 1235

CHR$ 13 es 'línea nueva'. Desplaza la posición del cursor al principio de la línea siguiente.

La televisión también emplea los códigos 16 a 23, que están detalladamente explicados en los dos capítulos siguientes. Todos los caracteres de control están relacionados en el apéndice A.

Empleando los códigos de los caracteres, podemos ampliar el concepto

de 'orden alfabético' para tratar cadenas que no sólo contengan letras sino cualquier carácter. Si en lugar de pensar en términos de alfabeto corriente (recuérdese que los ingleses no utilizan la ñ ni la ll, y sí la w), empleamos el alfabeto de 256 caracteres en el mismo orden de sus códigos, entonces el principio es exactamente el mismo. Por ejemplo,las siguientes cadenas están en orden alfabético en lo que al ZX Spectrum respecta (obsérvese la curiosa característica de que las letras minúsculas van después de las mayúsculas. Así la 'a' viene detrás de la 'Z'; también cuentan los espacios).

```
CHR$ 3+"ZOOLOGICO"
CHR$ 8+"CAZA DEL ZORRO"
" AAAAGH!"
"(algo entre paréntesis)"
"200"
"539 Ptas."
"ARMADILLO"
"Arcón"
"PRINT"
"Zulu"
"armadillo"
```

He aquí la regla para determinar el orden de dos cadenas. Primero, comparar el primer carácter de cada una. Si son distintos, entonces uno tendrá el código mayor que el otro y la cadena que irá en primer lugar será la del código más bajo. Si el carácter es el mismo, entonces se efectúa la misma comparación entre los segundos caracteres. En este proceso, si una de las cadenas se ejecuta antes que otra, es que dicha cadena es de código menor. En caso contrario, tienen el mismo código.

Las relaciones =, <, >, <=, >=, y <>se emplean en las cadenas del mismo modo que en los números : < equivale a 'viene antes' y > 'viene después'. Así pues,

```
"AA AAA" < "AAAAA"
"AAAAA"  >"AA AAA"
```

son expresiones correctas.

<= y >= actúan igual que con números. Así pues ,

```
"Esta cadena"<= "Esta cadena"
```

es verdad, pero..

```
"Esta cadena" < "Esta cadena"
```

es falso.

Experimente sobre todo ello entrando este programa que espera que le introduzcan dos cadenas para ponerlas en orden.

```
1Ø INPUT "Entre dos cadenas : ",a$,b$
2Ø IF a$>b$ THEN LET c$=a$:LET a$=b$:LET b$=c$
3Ø PRINT a$;" ";
4Ø IF a$<b$ THEN PRINT "<";:GO TO 6Ø
5Ø PRINT "="
6Ø PRINT " ";b$
7Ø GO TO 1Ø
```

Obsérvese como hemos introducido c$ en la línea 2Ø cuando intercambiamos a$ y b$.

```
LET a$=b$:LET b$=a$
```

no tendría los mismos efectos.

El siguiente programa establece gráficos predefinidos para las piezas de ajedrez.

P para peón
T para torre
C para caballo
A para alfil
R para rey
D para dama (reina)

Piezas de ajedrez

```
 5 LET b=BIN Ø11111ØØ:LET c=BIN ØØ111ØØØ:LET d=BIN ØØØ1ØØØØ
 1Ø FOR n=1 TO 6:READ p$:REM 6 piezas
 2Ø FOR f=Ø TO 7:REM 8 bytes para cada pieza
 3Ø READ a:POKE USR p$+f,a
 4Ø NEXT f
 5Ø NEXT n
 11Ø REM alfil
 115 DATA "a",Ø,d,BIN ØØ1Ø1ØØØ, BIN Ø1ØØØ1ØØ
 12Ø DATA BIN Ø11Ø11ØØ,c,b,Ø
 13Ø REM rey
 14Ø DATA "r",Ø,d,c,d
 15Ø DATA c,BIN Ø1ØØØ1ØØ,c,Ø
 16Ø REM torre
 165 DATA "t",Ø,BIN Ø1Ø1Ø1ØØ,b,c
 175 DATA c,b,b,Ø
 185 REM reina
 195 DATA "d",Ø,BIN Ø1Ø1Ø1ØØ,BIN ØØ1Ø1ØØØ,d
 200 DATA BIN Ø11Ø11ØØ,b,b,Ø
 2Ø5 REM peon
 21Ø DATA "p",Ø,Ø,d,c
 215 DATA c,d,b,Ø
 22Ø REM caballo
 225 DATA "c",Ø,d,c,BIN Ø1111ØØØ
 23Ø DATA BIN ØØØ11ØØØ,c,b,Ø
```

Obsérvese el empleo de Ø en lugar de BIN ØØØØØØØØ.

Cuando ejecute este programa, vea las piezas pasando al modo 'GRAPHICS'

Ejercicios.-

1.- Imagine el espacio para un símbolo (8x8) dividido en cuatro partes como un pastel. Entonces, cada cuarto puede ser blanco o negro por lo que tenemos 2x2x2x2=16 posibilidades. Encontrarlas en el set de caracteres..

2.- Ejecute el siguiente programa :

```
1Ø INPUT a
2Ø PRINT CHR$ a;
3Ø GO TO 1Ø
```

Si experimenta con él, encontrará que CHR$ a se redondea al entero más próximo; y si a no está entre Ø y 255, el programa se detiene con el informe B integer out of range (B entero fuera de escala).

3.- Cúal de estas cadenas es la más baja ?

"DEMONIO"
"demonio"

4.- Intente modificar el programa de obtener gráficos predefinidos empleando sentencias READ y DATA en lugar de INPUTS.

CAPITULO XV.- MAS ACERCA DE PRINT E INPUT

Resumen : CLS.Elementos PRINT :expresiones numéricas o de cadena.
Complementarios de PRINT : TAB, AT, , ; y '
Elementos INPUT : variables (numéricas o tipo cadena).
LINE Cadena variable
Elementos PRINT que no empiecen con una letra.
SCROLLING (Desplazamiento de la pantalla.)

Ya hemos visto el empleo de PRINT varias veces así que debe tener una idea bastante completa de su utilización. Las expresiones cuyos valores se imprimen se llaman elementos de PRINT y están separados por comas o punto y coma, llamados separadores de PRINT. Un elemento de PRINT también puede ser la nada absoluta, que es una manera de explicar qué sucede cuando usa dos comas en una fila.

Hay otras dos clases más de elementos de PRINT que se emplean para decir al computador donde debe imprimir en lugar de qué. Por ejemplo PRINT AT 11,16;"*", colocará una estrella en el centro de la pantalla.

AT línea, columna

mueve la posición PRINT (el lugar donde iba a imprimirse el siguiente carácter) hasta la línea y columna específica. Las líneas están numeradas de Ø (arriba) hasta 21 y las columnas de Ø (a la izquierda) a 31.

Véase tabla de la figura A, al final de este capítulo.

TAB columna

imprime suficientes espacios para mover la posición PRINT hasta la columna citada. Se mantiene en la misma línea o, si ello implicara salto de línea, iría a la siguiente. Vea que el computador divide el número entrado por 32 y toma el residuo. Así, TAB33 equivale a TAB 1.

Como ejemplo,

```
PRINT TAB 3Ø;1;TAB12;"Contiene";AT 3,1;"CAPITULO";TAB
24;"pag."
```

simularía la primera página de un libro.

Pruebe de entrar lo siguiente :

```
1Ø FOR n=Ø TO 2Ø
2Ø PRINT TAB 8*n;n;
3Ø NEXT n
```

que muestra el sistema por el que TAB divide por 32.

Para un ejemplo más vistoso cambie el 8 de la línea 2Ø por un 6.

Algunas puntualizaciones :

(i) Estos nuevos elementos quedan mejor terminados con punto y coma, como vimos anteriormente. Puede también usar comas (o nada, al final de la sentencia), pero ello equivale a cambiar el formato de la impresión después de haberla establecido con anterioridad cuidadosamente. Y no es, por cierto nada útil ni bello.

(ii) No se puede imprimir en las últimas dos líneas del pié de la pantalla (22 y 23) ya que están reservadas a las instrucciones, datos

de INPUT, informes, etc... Cuando nos referimos a la línea del fondo usualmente significa la 21.

(iii) Puede emplearse AT para imprimir incluso donde ya hay algo impreso. Esto significa que lo anterior desaparecerá para dar paso a la nueva impresión.

Otra instrucción relacionada con PRINT es la CLS. Esta sentencia "vacía" la pantalla entera, la deja en blanco algo así como CLEAR y RUN.

Cuando la impresión alcanza la parte inferior de la pantalla, empieza a desplazarse hacia arriba (scrolling). Lo comprobará si entra :

```
CLS;FOR n=1 TO 22:PRINT n:NEXT n
```

y a continuación

```
PRINT 99
```

varias veces.

Si el computador está imprimiendo gran cantidad de datos, tiene el cuidado suficiente de no hacer desaparecer por la parte superior de la pantálla ninguna información que no haya sido previamente examinada. Puede comprobarlo tecleando :

```
CLS:FOR n=1 TO 1ØØ:PRINT n:NEXT n
```

cuando ha llenado la pantalla, se detiene imprimiendo scroll? al pié de la misma. Puede ahora inspeccionar los primeros 22 números a su gusto. Cuando haya terminado, pulse 'y'(si) y el computador le presentará otra serie de números (del23 al 44). Realmente, cualquier tecla hace desplazarse la pantalla excepto n (no) STOP ó SPACE. Estas hacen detener el computador con el informe D BREAK-CONT repeats

La sentencia INPUT puede hacer bastante más de lo que hemos visto hasta ahora. Ha visto ya sentencias INPUT del tipo,

```
INPUT"Cuantos trozos quieres ?", partes
```

en donde el computador imprime la frase 'Cuantos trozos quieres ?' al pié de la pantalla y Vd. tiene que entrar el número pedido.

De hecho, una sentencia INPUT está formada por elementos y separaciones al igual que la sentencia PRINT, de manera que, 'Cuantos trozos quieres ?' y 'partes' son, ambas elementos del INPUT. Los elementos del INPUT son, en general, los mismos del PRINT, pero con unas importantes diferencias .

En primer lugar, un elemento adicional de INPUT es la variable que tiene que entrar (partes, en nuestro caso). La regla es que si una partícula o elemento INPUT empieza con una letra, debe tratarse de una variable cuyo valor ha de introducirse.

En segundo lugar, esto parecería indicar que no podemos imprimir valores de variables como parte de un epígrafe. No obstante puede obviar esta regla encerrando las variables entre paréntesis. Cualquier expresión que empiece con una letra debe encerrarse entre paréntesis si debe imprimirse como parte del epígrafe.

Cualquier clase de partícula PRINT que no se vea afectada por estas reglas es también una partícula INPUT. He aquí un ejemplo para ilustrar lo expuesto hasta ahora :

```
LET mi edad=INT(RND*1ØØ):INPUT("Tengo";mi edad;"."); Qué edad
tienes ?", tu edad
```

'mi edad' está entre paréntesis por lo que su valor se imprimirá.'tu edad' no está entre paréntesis por lo que tendrá que entrar su valor por el teclado.

Todo lo que escribe una sentencia INPUT, lo hace al pié de la pantalla y actúa con cierta independencia de la mitad superior de dicha pantalla. En particular sus líneas están numeradas con respecto a la línea superior de la mitad inferior de la pantalla, incluso si ésta se ha desplazado hacia arriba (lo que sucede si entra gran cantidad de datos de INPUT).

Para ver como AT funciona en las sentencias INPUT, pruebe lo siguiente

```
1Ø INPUT "Esta es la línea 1.",a$;AT Ø,Ø;"Esta es la linea Ø.",
   a$;AT 2,Ø;"Esta es la linea 2.",a$;AT 1,Ø;"Esta es todavia
   la linea 1.",a$
```

(pulse ENTER cada vez que el computador se detenga). Cuando'Esta es la línea 2' se imprima,la parte inferior de la pantalla se desplaza hacia arriba para dejarle espacio; pero la numeración se desplaza también hacia arriba de manera que las líneas de texto siguen conservando sus números.

Ahora, pruebe esto :

```
1Ø FOR n=Ø TO 19:PRINT AT n,Ø;n;:NEXT n
2Ø INPUT AT Ø,Ø;a$;AT1,Ø;a$;AT 2,Ø;a$;AT 3,Ø;a$;
   AT 4,Ø;a$;AT 5,Ø;a$;
```

Dado que la parte baja de la pantalla va subiendo hacia arriba, la parte superior permanece inalterable hasta que dicha parte inferior intenta escribir en la misma línea de la posición PRINT. Entonces la parte superior se desplaza hacia arriba (hace scroll) para evitarlo.

Otro refinamiento de la sentencia INPUT que no hemos visto todavía es la llamada LINE y se trata de una manera alternativa de introducir variables de cadena. Si escribe LINE inmediatamente antes del nombre de una cadena variable del modo

```
INPUT LINE a$
```

entonces el computador no le dará las comillas que normalmente se emplean en las variables de cadena, aunqué actuará como si estuvieran. Así pues, entre

```
gato
```

como dato de INPUT, a$ tendrá el valor gato. Dado que las comillas no aparecen en la cadena, no podrá borrarla ni entrar otra en su lugar como dato de INPUT. Recuerde que LINE no puede emplearse en variables numéricas.

Los caracteres de control CHR$22 y CHR$23 tienen efectos parecidos a AT y TAB. Son bastante curiosos como caracteres de control porqué cada vez que uno de ellos es enviado a la pantalla para ser impreso, debe ir seguido por dos caracteres adicionales que no tienen su cometido habitual: son tratados como números (sus códigos) para especificar la línea y la columna (por AT) o la posición de tabulación (por TAB). Prácticamente siempre encontrará más sencillo emplear AT y TAB en la forma habitual que emplea los caracteres de control, pero éstos pueden ser útiles en determinadas circunstancias. El control caracter de AT es CHR$22. El primer carácter después del mismo especifica el número de línea y el segundo el número de columna, así que

```
PRINT CHR$ 22+CHR$ 1+CHR$ c;
```

equivale exactamente a

```
PRINT AT 1,c;
```

Así pues, incluso aunque CHR$ 1 o CHR$ c tuvieran normalmente un significado distinto (por ejemplo si c=13), al poner CHR$ 22 delante quedan invalidados.

El caracter de control de TAB es CHR$ 23 y los dos caracteres que le siguen se emplean para dar un número entre Ø y 65535 que representa lo mismo (el mismo número) que hubiera empleado usando TAB.

```
PRINT CHR$ 23+CHR$ a+CHR$ b;
```

equivale a

```
PRINT TAB a+256*b;
```

Puede emplear POKE para hacer que el computador se detenga preguntándole a Vd, scroll? tecleando

```
POKE 23692,255
```

periódicamente. Después de ello se desplazará la pantalla hacia arriba 255 veces antes de pararse de nuevo con la pregunta scroll ?. Como ejemplo pruebe

```
10 FOR n=Ø TO 1ØØØØ
20 PRINT n;POKE 23692,255
30 NEXT n
```

y observe lo que sucede en la pantalla.

Ejercicios.-

1.- Pruebe este programa con varios chicos para ver como están de matemáticas :

```
1Ø LET m$=""
2Ø LET a=INT(RND*12)+1:LET b=INT(RND*12)+1
3Ø INPUT(m$)"cuanto es ";(a)"*";(b);"?";c
1ØØ IF c=a*b THEN LET m$="Correcto.":GO TO 2Ø
11Ø LET m$="Mal. Prueba de nuevo,":GO TO 3Ø
```

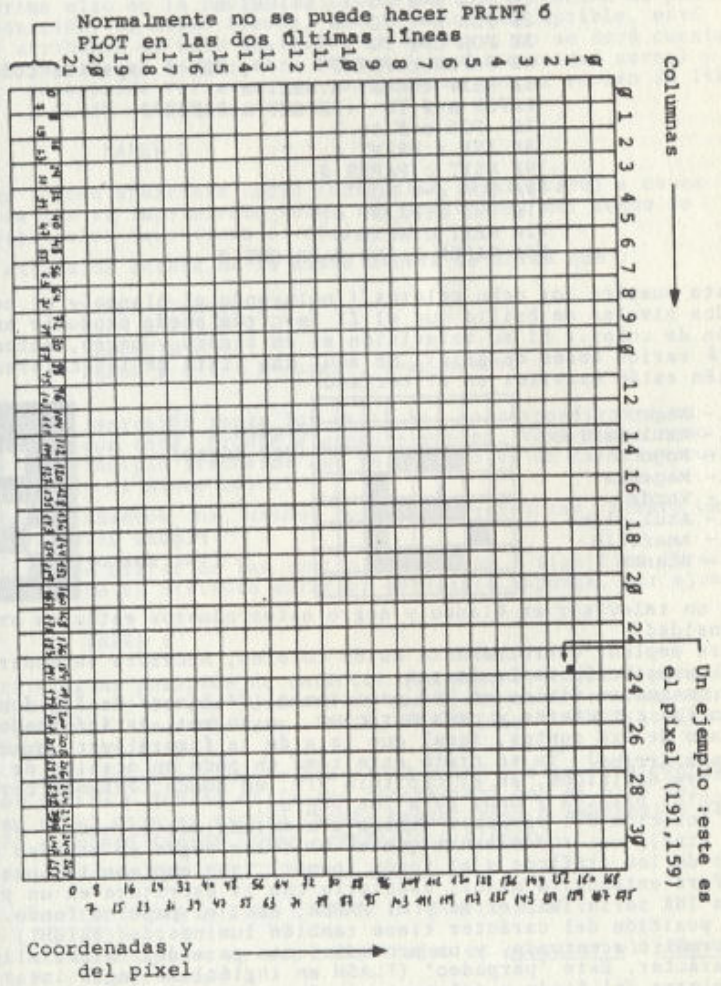
Si son unos chicos listos, se las apañarán para no tener que calcular ellos mismos. Simplemente, si el computador les pide la respuesta de 2*3 (p.ejemplo), sólo tienen que entrar 2*3 para que sea correcto.

Una manera de evitar esta pequeña trampa es hacer que entren cadenas en lugar de números. Sustituya c en la línea 3Ø por c$ y, en la línea 1ØØ, por VAL c$, e inserte una línea como ésta.

```
4Ø IF c$<>STR$ VAL c$ THEN LET m$="Entrarlo correctamente, como un número.": GO TO 3Ø.
```

Esto les confundirá. No obstante, tras algunos intentos, quizás descubran el truco de borrar las comillas y entrar STR$(2*3)(Si es así "chapeau" para las criaturas). Para evitar esto, puede reemplazar c$ de la línea 3Ø por LINE c$.

FIGURA A (ver texto)



CAPITULO XVI.- LOS COLORES

Resumen : INK, PAPER, FLASH, BRIGHT, INVERSE, OVER
BORDER.

Pruebe este programa :

```
1Ø FOR m=Ø TO 1 :BRIGHT m
2Ø FOR n=1 TO 1Ø
3Ø FOR c=Ø TO 7
4Ø PAPER c:PRINT"    ";:REM 4 espacios coloreados
5Ø NEXT c:NEXT n:NEXT m
6ØFOR m=Ø TO 1:BRIGHT m:PAPER 7
7Ø  FOR c=Ø to 3
8Ø INK c:PRINT c;" ";
9Ø NEXT c:PAPER Ø
1ØØ FOR c=4 TO 7
11Ø INK c:PRINT c;" ";
12Ø NEXT c:NEXT m
13Ø PAPER 7:INK Ø:BRIGHT Ø
```

Esto muestra los ocho colores (incluyendo el blanco y el negro) y los dos niveles de brillo que el ZX Spectrum puede producir en una tele visión de color.(Si su televisión es en blanco y negro, entonces ob tendrá varios tonos de gris). He aquí una lista de los colores, que también están escritos en el teclado.

Ø.- Negro
1.- Azul marino
2.- Rojo
3.- Magenta
4.- Verde
5.- Azul claro
6.- Amarillo
7.- Blanco

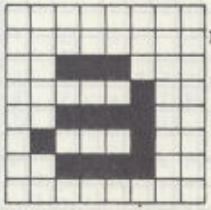


Figura 1
(Ver texto)

Figura 2
(Ver texto)

En un televisor en blanco y negro estos números están en orden de luminosidad.

Para emplear correctamente estos colores, necesita entender un po co la composición de la imagen.

La imagen se divide en 768 posiciones (24 líneas de 32) donde se im primen los caracteres y cada carácter , a su vez, está formado por un cuadrado de 8x8 puntos, igual que la a de la figura (ver figura 1 dibujada más arriba) Ya se trató este tema un poco en ocasión de los grá ficos pre-definidos, en el capítulo XIV, en donde teníamos ceros para los puntos blancos y unos para los negros.

La posición del carácter conlleva asociada dos colores el texto o tinta (ink), color dominante que es el color negro en el caso de los puntos de los gráficos y el fondo (paper), que emplean los puntos blan cos. Para entenderlo mejor, asociarlo con la escritura en un papel blan co La INK sería las letras y el PAPER, eso, el papel o fondo.

La posición del carácter tiene también luminosidad(BRIGHT) que puede ser normal o acentuada, y un procedimiento para determinar el'parpadeo' del carácter. Este 'parpadeo' (FLASH en inglés) se logra invirtiendo los colores del fondo y del texto de forma intermitente Todo ello está convenientemente codificado mediante números de manera que una posición de carácter tiene :

a) un cuadro de 8x8 puntos (ceros y unos) para definir la forma del carácter, con ceros para el PAPER(fondo) y unos para INK (texto o tinta).

b) colores para la tinta y el fondo, con números comprendidos entre el Ø y el 7.
c) luminosidad : Ø para normal y 1 para acentuada (o incrementada).
d) un número de 'parpadeo' (flashing): Ø para fijo y 1 para 'parpadeo'.

Nótese que dado que los colores de tinta y fondo ocupan una única posición de carácter, no es posible tener más de dos colores en un mis mo bloque de 64(8x8) puntos. Lo mismo es aplicable a los números de luminosidad y parpadeo : se refieren al bloque entero y no a cada punto por separado. Los colores, luminosidad y parpadeo de una posición de terminada se llaman atributos.

Cuando imprime algo en la pantalla, cambia la configuración de pun tos de esta posición; al mismo tiempo, aunque menos perceptible, está cambiando los atributos de esta posición. Al empezar, no se dará cuenta ya que todo se imprime en negro sobre blanco (con luminosidad normal y sin parpadeo), pero puede variar estas condiciones con el empleo de INK, PAPER, BRIGHT, y FLASH. Entre :

```
PAPER 5
```

e imprima algo : todo aparecerá sobre fondo cian (azul claro) a causa de que a medida que va imprimiendo todas las posiciones del fondo se cambiarán a este color (que tiene el código 5)..

Los otros atributos actúan de la misma manera de forma que :

```
PAPER n°entre Ø y 7
INK  n°entre Ø y 7
BRIGHT Ø ó 1 }
FLASH Ø ó 1  } Ø=normal, 1=en marcha
```

hará que cualquier impresión tenga los atributos correspondientes en cualquier posición que esté. Pruebe algunos ejemplos. Verá ahora como funciona todo al principio (recuerde que un espacio es un carácter cuyo fondo y texto tienen el mismo color).

Existen algunos números que, usados con estas sentencias, tienen un efecto menos aparente.

El número 8 puede usarse en las cuatro sentencias y significa 'trans parente' de forma que el atributo anterior permanece.Suponga, por ejem plo :

```
PAPER 8
```

No existirá ninguna posición de carácter que tome el color 8 porqué no existe dicho color ; lo que sucederá es que dicho carácter mantendrá el color que ya tenía sin cambiarse. INK 8, BRIGHT 8 y FLASH 8 actúan de la misma forma citada.

El número 9 sólo puede emplearse con PAPER e INK y significa'contras te'. El color (texto o fondo) utilizado contrastará con el otro vol viéndolo blanco si el otro es oscuro (azul, negro, rojo o magenta) y ne gro si el otro es claro (verde, cian, amarillo o blanco).

Compruébelo entrando

```
INK 9:FOR c=Ø TO 7:PAPER c:PRINT c:NEXT c
```

Una manera más descriptiva de sus habilidades es ejecutar el progra ma del principio y luego hacer

```
INK 9:PAPER 8:PRINT AT Ø,Ø;:FOR n=1 TO 1ØØØ:PRINT n;:NEXT n
```

El color del texto está previsto aquí que contraste siempre con el fondo anterior en cada posición.
El color en la televisión se basa en el ciertamente curioso hecho

de que el ojo humano únicamente ve tres colores (los colores primarios, azul, rojo y verde). Los restantes son meras mezclas de éstos. Por ejemplo , el magenta se logra mezclando azul y rojo, por lo que su código, el 3, es la suma de los códigos azul y rojo .

Para ver como combinan entre sí los ocho colores, imagine tres proyectores rectangulares, con los colores azul, rojo y verde. Los demás colores son mezclas de éstos. Si dichos proyectores se solapan entre sí verá sus mezclas de colores tal y como muestra el siguiente programa (observe que los espacios de texto se obtienen usando cualquier SHIFT con 8 en modo G).

```
10 BORDER 0:PAPER 0:INK 7:CLS
20 FOR a=1 TO 6
30 PRINT TAB 6;INK 2;"##################":REM 18
   cuadrados negros
40 NEXT a
50 LET línea de datos=200
60 GO SUB 1000
70 LET línea de datos=210
80 GO SUB 1000
90 STOP
200 DATA 2,3,7,5,4
210 DATA 2,2,6,4,4
1000 FOR a=1 TO 6
1010 RESTORE línea de datos
1020 FOR b=1 TO 5
1030 READ c:PRINT INK c;"######";:REM 6 cuadrados
1040 NEXT b:PRINT:NEXTa
1050 RETURN
```

Nota # equivale o debe ser espacio negativo o cuadro negro

Existe una función llamada ATTR que averigua los atributos de una posición determinada de pantalla. Es una función bastante complicada por lo que ha sido relegada al final de este capítulo.

Existen dos sentencias más, INVERSE y OVER que controlan, no los atributos sino la configuración de puntos impresa en la pantalla. Emplean los números 0 (para desactivado) y 1 (en marcha) al estilo de FLASH y BRIGHT pero son sus únicas posibilidades. Si escribe INVERSE 1 entonces la configuración de puntos impresa será la inversa de la habitual, es decir, el negativo : los puntos del fondo serán reemplazados por los del texto y viceversa. De este modo, la letra a se imprimiría igual que en la figura 2 del principio de este capítulo).

Si, como al empezar, tenemos texto negro sobre fondo blanco, entonces esta a aparecerá como blanca sobre negro, pero continuamos teniendo negro sobre blanco en dicha posición. Son los puntos los que han cambiado.

La sentencia

```
OVER 1
```

actúa como una cierta sobre-impresión . Normalmente, cuando algo es escrito en una posición, anula completamente lo que había anteriormente; con esta instrucción el nuevo carácter simplemente se superpone sobre el otro (véase ejercicio 3). Esto puede ser realmente útil para escribir caracteres compuestos, tales como letras con acento, como en este programa para imprimir letras germanas- una 'o' con diéresis encima-. (Pulse NEW ante todo).

```
10 OVER 1
20 FOR n=1 TO 32
30 PRINT "o";CHR$ 8;"""";
40 NEXT n
```

De forma parecida podríamos tratar de hacer nuestra ñ española. Desgraciadamente no existe ningún caracter ASCII que podamos emplear para hacer el guión superior. Intente entrar la línea 30 del siguiente modo

y le saldrá una ñ algo apañadita

```
30 PRINT"n";CHR$8;CHR$39:
```

nótese que CHR$8 retrocede un espacio y que CHR$ 39 es el código del acento (que no es sino un mal remedo del guión de la ñ, lo siento)

Existe otra manera de emplear INK, PAPER, etc. que encontrará probablemente más práctica en lugar de usarlos como sentencias. Puede colocarlas como partículas de la sentencia PRINT (seguidas por ;) y haciendo exactamente el mismo cometido que harían actuando por su cuenta con la salvedad de que su efecto es temporal durando solamente lo mismo que la sentencia PRINT que las contiene. Entonces, si entra

```
PRINT PAPER 6;"x";:PRINT "y"
```

solamente tendremos la x en color amarillo.

INK y el resto de las sentencias, cuando se emplean como tales no afectan a los colores de la parte inferior de la pantalla, donde se entran los comandos y los datos de INPUT. Esta parte inferior usa el color del contorno (BORDER) para el fondo y el código 9 para cualquier color que contraste, no parpadea ni tiene luminosidad acentuada. Puede cambiar el color del contorno (BORDER) para obtener cualquiera de los 8 disponibles empleando la sentencia

```
BORDER color
```

Cuando esté entrando datos de INPUT, sigue la regla de contrastar el texto con el fondo coloreado como el contorno, pero puede cambiar el color de las frases escritas por el computador usando INK y PAPER (etc.) en la sentencia INPUT, igualmente que si estuviera en una sentencia PRINT Su efecto dura hasta el final de la sentencia o hasta que algún dato de INPUT tenga qùe ser introducido, lo que primero ocurra. Pruebe :

```
INPUT FLASH 1;INK 1;"Cual es su numero ?";n
```

Hay una manera de cambiar los colores mediante el empleo de caracteres de control, de forma parecida a los caracteres de control de AT y TAB en el capítulo XV.

```
CHR$ 16 corresponde a INK
CHR$ 17 corresponde a PAPER
CHR$ 18 corresponde a FLASH
CHR$ 19 corresponde a BRIGHT
CHR$ 20 corresponde a INVERSE
CHR$ 21 corresponde a OVER
```

cada uno de ellos seguido por un carácter que indique el código del color : así (p.e.)

```
PRINT CHR$ 16+CHR$ 9;..
```

equivale a

```
PRINT INK 9;..
```

Realmente no valdría la pena preocuparse por estos caracteres de control ya que puede usar directamente los colores. No obstante puede resultar muy útil para colocarlos en programas : podrá tener diferentes secciones diferenciadas por colores lo cual, aparte de útil puede resultar algo más elegante. Debe colocarlos después de una línea de programa o se perderán.

Para integrarlos en un programa, debe introducirlos por el teclado usando el modo extendido con los dígitos.

Los dígitos Ø al 7 determinan el color correspondiente (al texto si tenemos pulsado CAPS SHIFT,o al fondo en caso contrario). Más concretamente, si estamos en modo E y pulsamos un dígito (el 6 del amarillo p.e. pues debe estar entre Ø y 7 , no valen 8 y 9), entonces se insertan dos caracteres : primero CHR$ 17 para PAPER y CHR$6indicando 'el amarillo'. Si estaba apretando al mismo tiempo CAPS SHIFT cuando pulsó el dígito, tendrá CHR$16 que equivale a INK en lugar de CHR$17.

Dado que hay dos caracteres, es posible que observe efectos curiosos cuando intente borrarlos. Debe apretar DELETE dos veces y, después de la primera vez posiblemente obtenga un signo de interrogación e, incluso algunas letras sin sentido. No se preocupe: apriete DELETE de nuevo.

◆ y ◆ pueden también comportarse de manera peculiar cuando el cursor se desplaza más allá de los caracteres de control.

Todavía en modo extendido (cursor E), tenga en cuenta que :

8 da CHR$ 19 y CHR$ Ø para luminosidad normal
9 da CHR$ 12 y CHR$ 1 para luminosidad acentuada
CAPS SHIFT con 8 da CHR$ 18 y CHR$ Ø sin parpadear
CAPS SHIFT con 9 da CHR$ 19 y CHR$ 1 con parpadeo.

Hay un par de funciones más en modo ordinario (L). :

CAPS SHIFT con 3 da CHR$ 2Ø y CHR$ Ø en caracteres normales.
CAPS SHIFT con 4 da CHR$ 2Ø y CHR$ 1 en caracteres inversos.

Para resumir ver tabla de la Figura 3 al final del capítulo que muestra todas las posibilidades de la fila superior de teclas.

La función ATTR tiene la forma

ATTR (línea columna)

Sus dos argumentos son la línea y la columna que ya habíamos usado en la partícula AT, y su resultado es un número que indica los colores y demás características del carácter definido por la fila y la columna. Puede emplearse esta función tranquilamente en expresiones como cualquier otra función.

El número resultante es la suma de cuatro números distintos que son:

1°) 128 si el caracter parpadea, Ø si no.
2°) 64 si el carácter brilla, Ø si es normal
3°) 8 x el código del color del fondo
4°) el número código del texto.

Por ejemplo, si la posición del carácter está parpadeando con brillo normal, con fondo amarillo y texto azul, entonces los cuatro números que tenemos que sumar serán 128, Ø, 8*6=48 y 1, siendo el resultado 177.

Pruebe esto entrando :

```
PRINT AT Ø,Ø;FLASH 1;PAPER 6;INK 1;" ";ATTR(Ø,Ø)
```

Ejercicios.-

1.- Pruebe

```
PRINT "B";CHR$ 8;OVER 1;"/";
```

Donde el / ha cortado a la B ha dejado un punto blanco. Esta es la manera de sobreimprimir que tiene el ZX Spectrum :dos fondos o dos textos dan un fondo, uno de ellos da un texto. Ello tiene la interesante propiedad de que si sobreimprime por dos veces un carácter con otro cualquiera, vuelve a tener el mismo carácter original. Si ahora teclea :


```
PRINT CHR$ 8;OVER 1;"/"
```

¿porqué recupera una B sin mácula alguna ?.

2.- Teclee :

```
PAPER Ø:INK Ø
```

¿verdad que no afectan a la parte inferior de la pantalla ?. Ahora entre :

```
BORDER Ø
```

y vea lo que el Spectrum hace por usted !

Ejecute el siguiente programa :

```
1Ø POKE 22527+RND*7Ø4,RND*127
2Ø GO TO 1Ø
```

No le preocupe como trabaja este programa ; está cambiando los colores de los cuadros de la pantalla de TV y el RND asegura que ello se efectuará aleatoriamente. Las rayas diagonales que aparezcan esporádicamente, son una manifestación de la configuración oculta de RND que hace que, en realidad, sea pseudoaleatoria en lugar de auténtica.

4.- Entre o cargue (LOAD) los caracteres de las piezas de ajedrez del capítulo XIV y luego introduzca este programa que muestra la imagen de un tablero de ajedrez, usando los caracteres citados.

```
5 REM Dibujo del tablero vacío
1Ø LET bb=1:LET bw=2:REM azul y rojo
15 PAPER bw:INK bb:CLS
2Ø PLOT 79,128:REM borde
3Ø DRAW 65,Ø:DRAW Ø,-65
4Ø DRAW -65,Ø:DRAW Ø,65
5Ø PAPER bb
6Ø REM tablero
65 FOR n=Ø TO 3:FOR m=Ø TO 3
7Ø PRINT AT 6+2*n,11+2*m;" "
75 PRINT AT 7+2*n,1Ø+2*m;" "
8Ø NEXT m:NEXT n
85 PAPER 8
9Ø LET pw=6:LET pb=5:REM colores piezas
95 DIM b$(8,8):REM posiciones piezas
98 REM establecer posiciones iniciales
1ØØ LET b$(1)="tcadract"
1Ø5 LET b$(2)="pppppppp"
11Ø LET b$(7)="PPPPPPPP"
115 LET b$(8)="TCADRACT"
12Ø REM dibujar tablero
122 FOR n=1 TO 8:FOR m=1 TO 8
125 LET bc=CODE b$(n,m):INK pw
13Ø IF bc=CODE" "THEN GO TO 35Ø:
    REM espacio
135 IF bc>CODE"Z"THEN INK pb:
    LET bc=bc-32:REM minúsculas
    para color negro
14Ø LET bc=bc+79: REM gráficos
145 PRINT AT 5+n,9+m;CHR$ bc
15Ø NEXT m:NEXT n
2ØØ PAPER 7:INK Ø
```

FIGURA 3 (ver texto)

MODO	SHIFT										
E	SYMBOL	DEF	FN	LINE	OPEN	CLOS	MOVE	ERASE	POINT	CAT	FOR MAT
	CAPS	INK azul	INK rojo	INK mag.	INK verde	INK cian	INK amar.	INK blanc	FLASH off	FLASH on	INK negro
	NADA	PAPEL azul	PAPEL rojo	PAPEL mag.	PAPEL verde	PAPEL cian	PAPEL amar.	PAPEL blan	BRILL norm.	BRILL extra	PAPEL negro
G	CUAL QUIERA	■	■	■	■	■	■	■	■	SALIDA GRAF.	BO RRADO
	NADA	■	■	■	■	■	■	■	□	SALIDA GRAF.	BO RRADO
K, L ó C	CAPS	EDIT	CAPS LOCK	TRUE VIDEO	INV. VIDEO	⇦	⇩	⇧	⇨	MODO GRAF.	BO RRADO
	SYMBOL	!	@	#	$	%	&	'	(	)	_
	NADA	1	2	3	4	5	6	7	8	9	Ø

CAPITULO XVII.- GRAFICOS

Resumen : PLOT, DRAW, CIRCLE
POINT, pixels

En este capítulo veremos como dibujar imágenes en el ZX Spectrum. La parte de la pantalla que podemos usar tiene 22 líneas y 32 columnas, haciendo un total de 22x32=704 posiciones. Como recordará del capítulo XVI, cada una de estas posiciones está formada por 8x8 cuadrados de puntos, siendo estos puntos llamados pixels.

Un pixel está definido por dos números, sus coordenadas. La primera, la coordenada x indica la separación del margen izquierdo de la pantalla. La segunda, la coordenada y indica la distancia que lo separa del fondo de la pantalla. Dichas coordenadas se escriben normalmente dentro de paréntesis, así (0,0), (255,0), (0,175) y (255,175) representan las esquinas inferior izquierda, inferior derecha, superior izquierda y superior derecha respectivamente.

La sentencia PLOT coord.x, coord.y

dibuja en pantalla el pixel de estas coordenadas, de modo que el siguiente programa

```
10 PLOT INT(RND*256),INT(RND*176):INPUT a$:GO TO 10
```

dibuja un punto aleatorio cada vez que pulse ENTER(igual que la viruela)!
He aquí un programa bastante más interesante. Dibuja la función SIN (seno) para valores entre 0 y 2π .(la clásica senoide).

```
10 FOR n=0 TO 255
20 PLOT n,88+80*SIN(n/128*PI)
30 NEXT n
```

Este otro programa dibuja un gráfico de la función SQR (raiz cuadrada) entre 0 y 4 (una semiparábola).

```
10 FOR n=0 TO 255
20 PLOT n,80*SQR(n/64)
30 NEXT n
```

Advierta que las coordenadas del pixel son bastante distintas de las empleadas por la partícula AT. Consulte el diagrama del capítulo XV si necesita trabajar con pixels, líneas y columnas (pag 102 del manual original).

Para ayudarle en sus gráficos, el computador puede dibujar líneas rectas, círculos y partes de círculo, empleando las sentencias DRAW y CIRCLE.

La sentencia DRAW para dibujar (draw en inglés) una línea recta adopta la forma

DRAW x,y

El punto de partida de la línea es el pixel en donde quedó la última sentencia PLOT, DRAW o CIRCLE (ésta es la llamada posición PLOT; RUN, CLEAR, CLS y NEW la reponen a la esquina inferior izquierda, a (0,0)), y el punto final está x pixels a la derecha e y pixels más arriba. La sentencia DRAW por sí sola determina la longitud y dirección de la línea pero no su punto de partida.

Experimente con algunas sentencias PLOT y DRAW, por ejemplo,


```
PLOT 0,100:DRAW 80,-35
PLOT 90,150:DRAW 80,-35
```

Observe que los números de la sentencia DRAW pueden ser negativos, al contrario que la PLOT que sólo acepta positivos.

Puede también hacer PLOT y DRAW en color, aunqué tiene que tener en cuenta que los colores siempre cubren la totalidad del carácter y no puede definirlo para pixels individualmente. Cuando se activa un pixel adopta el color del texto (o tinta) determinado, conservando el resto del carácter el color que tenía en aquel momento. Compruébelo con el siguiente programa :

```
10 BORDER 0:PAPER 0:INK 7:CLS:REM ennegrecer la pantalla
20 LET x1=0:LET y1=0:REM principio de línea
30 LET c=1:REM para color texto, empezando por azul
40 LET x2=INT(RND*256):LET y2=INT(RND*176):REM final aleatorio de la línea
50 DRAW INK c;x2-x1,y2-y1
60 LET x1=x2:LET y1=y2:REM la próxima línea empieza donde termina la anterior
70 LET c=c+1:IF c=8 THEN LET c=1:REM nuevo color
80 GO TO 40
```

Las líneas parecen hacerse cada vez más anchas a medida que se ejecuta el programa y esto es a causa de que una línea cambia los colores de los pixels de texto de todas las posiciones que recorre. Observe que puede incluir partículas PAPER, INK, FLASH, BRIGHT, INVERSE y OVER en las sentencias PLOT y DRAW como si se trataran de PRINT e INPUT. Estas partículas van entre la instrucción y las coordenadas y terminan con comas o punto y coma..
Una característica adicional de DRAW es que puede emplearlo para dibujar partes de circunferencia además de líneas rectas, usando un número suplementario que especifica un ángulo de giro ; la forma es

DRAW x,y,a

donde x e y se emplean para definir el punto final de la línea como antes y a es el número de radianes que debe girar en su recorrido; si a es positivo gira hacia la izquierda y si es negativo, hacia la derecha. Otra forma de considerar a es como indicadora de la fracción de círculo que recorre : un círculo completo equivale a 2π radianes así que, a=π dibujará un semicírculo, a=0.5*π un cuarto de círculo, etc..

Por ejemplo, supongamos a=π . Entonces, para cualquier valor que adopten x e y, se dibujará un semicírculo. Ejecute :

```
10 PLOT 100,100:DRAW 50,50,PI
```

que mostrará algo como esto

El dibujo empieza en dirección sureste y, cuando termina, va en dirección noroeste. Entre uno y otro punto ha girado 180°en redondo, ó π radianes (el valor de a).

Ejecute este programa varias veces, reemplazando PI por otras expresiones,p.e.-PI, PI/2, 3*PI/2, PI/4, 1, Ø, etc.

La última sentencia de este capítulo es CIRCLE (círculo) que dibuja una circunferencia completa. Hay que definir las coordenadas del centro y el radio de la forma,

```
CIRCLE coord.x,coord.y,radio
```

Al igual que con PLOT y DRAW, puede colocar las partículas de color de su elección al principio de la sentencia CIRCLE.

La función POINT (punto) le dice cuando un pixel es del color texto o fondo. Tiene dos argumentos, las coordenadas del pixel (encerradas en paréntesis); su resultado es Ø si el pixel es color fondo y 1 si es de color texto o tinta. Pruebe

```
CLS:PRINT POINT (Ø,Ø):PLOT Ø,Ø:PRINT POINT (Ø,Ø)
```

Teclee

```
PAPER 7:INK Ø
```

y averigüemos como trabajan INVERSE y OVER en una sentencia PLOT. Ambas sólo afectan al pixel fundamental y no al resto de la posición.Normalmente no actúan (Ø) en una sentencia PLOT así que debe mencionarlas específicamente si desea que actúen (1).

He aquí una lista de sus posibilidades :

PLOT; ésta es la forma usual. Coloca un punto de tinta (texto),p.e. el pixel para mostrar el color de texto.

PLOT INVERSE 1; coloca un punto 'anulador' de tinta, p.e. coloca el pixel que muestra el color del fondo

PLOT OVER 1; Esta forma cambia el pixel anterior fuera cual fuera : así si era del color texto cambia a fondo y viceversa.

PLOT INVERSE 1;OVER 1; deja el pixel exactamente como estaba, pero observe que cambia la posición PLOT, así que puede emplearia simplemente para efectuar esta operación.

Como otro ejemplo de empleo de OVER, llene la pantalla escribiendo en blanco y negro, y luego entre

```
PLOT Ø,Ø:DRAW OVER 1;275,175
```

esto dibujará una línea con bastante precisión a pesar de tener fallas cada vez que encuentre algo impreso en su camino. Ahora, repita de nuevo el texto anterior. La línea desaparecerá sin dejar rastro. Esta es la gran ventaja de OVER1. Si hubiera dibujado la línea usando

```
PLOT Ø,Ø:DRAW 255,175
```

y borrado con

```
PLOT Ø,Ø:DRAW INVERSE1;255,175
```

habría también borrado parte de lo escrito debajo.

Ahora pruebe

```
PLOT Ø,Ø:DRAW OVER 1;25Ø,175
```

y trate de borrarlo con

```
DRAW OVER 1;-25Ø,-175
```

Lo logrará defectuosamente puesto que los pixels que la línea emplea en el retorno no son los mismos que empleó a la ida. Debe borrar una línea

exactamente de la misma manera que la dibujó (misma dirección).

Una forma de conseguir colores no standarts es introduciendo dos colores base en una misma posición usando un gráfico predefinido. Ejecute este programa :

```
1ØØØ FOR n=Ø TO 6 STEP 2
1Ø1Ø POKE USR "a" +n,BIN Ø1Ø1Ø1Ø1:POKE USR "a"+n+1,
     BIN 1Ø1Ø1Ø1Ø
1Ø2Ø NEXT n
```

el cual nos dará el gráfico que define un tablero de ajedrez. Si imprime este carácter (modo Gráficos) con la tecla a en rojo sobre fondo amarillo, obtendrá un color naranja bastante correcto.

Ejercicios .-

1.- Experimente con las partículas PAPER, INK, FLASH y BRIGHT dentro de una sentencia PLOT. Estas son las partes que afectan la totalidad de la posición de carácter que contiene el pixel. Normalmente es como si la sentencia PLOT en si misma, fuera :

```
PLOT PAPER 8;FLASH 8; BRIGHT 8;...
```

y solo se ve alterado el color del texto cuando se coloca algo ahí, pero puede cambiarlo a voluntad.

Sea especialmente cuidadoso cuando use colores con INVERSE 1, porqué obliga al pixel a mostrar el color del fondo, pero cambia el color del texto y no siempre será lo que Vd. esperaba.

2.- Intente dibujar círculos empleando SIN y COS (si ha leido el capítulo IX, aplique sus conocimientos). Ejecute esto :

```
1Ø FOR n=Ø TO 2*PI STEP PI/18Ø
2Ø PLOT 1ØØ+8Ø*COS n,87+8Ø*SIN n
3Ø NEXT n
4Ø CIRCLE 15Ø,87,8Ø
```

Observe que la sentencia CIRCLE es mucho más rápida,aunque menos precisa.

3.- Pruebe :

```
CIRCLE 1ØØ,87,8Ø : DRAW 5Ø,5Ø
```

y constatará que la sentencia CIRCLE deja la posición PLOT en un lugar bastante indefinido, siempre en algún lugar de la mitad derecha del círculo. Necesitará colocar una sentencia PLOT después de CIRCLE siempre que tenga que continuar dibujando.

4.- He aquí el programa para trazar la gráfica de casi cualquier función. Primero le insta a introducir un número n; luego trazará la gráfica para valores de -n a n. A continuación le pedirá que entre la función deseada, como una cadena. Dicha cadena debe ser una expresión que emplee x como argumento de la función.

Ejecute este programa y, como ejemplo, introduzca 1Ø como valor de n y la función 1Ø*TAN x. Trazará una gráfica de tangente de x para valores de x de -1Ø a +1Ø.

```
1Ø PLOT Ø,87:DRAW 255,Ø
2Ø PLOT 127,Ø:DRAW Ø,175
3Ø INPUT n,e$
4Ø LET t=Ø
45 FOR f=Ø TO 255
5Ø LET x=(f-128)*n/128:LET y=VAL e$
6Ø IF ABS y> 87 THEN LET t=Ø:GO TO 1ØØ
7Ø IF NOT t THEN PLOT f,y+88:LET t=1:GO TO 1ØØ
8Ø DRAW 1,y-vieja y
1ØØ LET vieja y =INT(y+.5)
11Ø NEXT f
```

CAPITULO XVIII.- MOVIMIENTO

Resumen : PAUSE, INKEY$, PEEK

Bastante a menudo querrá lograr que el programa tenga una determinada duración. Para ello,la sentencia PAUSE (pausa) le será particularmente de utilidad.

PAUSE n

interrumpe el cómputo y congela el contenido de la pantalla durante n ciclos de cuadro de TV.(50 veces por seg. en Europa y 60 en EEUU).n debe ser menor de 65535, que equivale a 22 minutos y que es el máximo tiempo de pausa controlada; si hacemos n=0 estonces significa 'pausa infinita'.

Una pausa siempre puede acortarse pulsando una tecla (CAPS SHIFT y SPACE también actúan como BREAK en este caso) cualquiera. Debe pulsar dicha tecla, si quiere, a partir de que haya comenzado la pausa.

El siguiente programa simula la actuación de la aguja segundera de un reloj.

```
10 REM Dibujamos el reloj
20 FOR n=1 TO 12
30 PRINT AT 10-10*COS(n/6*PI),16+10*SIN(n/6*PI);n
40 NEXT n
50 REM Ponemos en marcha el reloj
60 FOR t=0 TO 200000:REM t es el tiempo en segundos
70 LET a=t/30*PI:REM a es el ángulo del segundero en
   radianes
80 LET sx=80*SIN a:LET sy=80*COS a
200 PLOT 128,88:DRAW OVER 1;sx,sy:REM traza el segundero
210 PAUSE 42
220 PLOT 128,88:DRAW OVER 1;sx,sy:REM borra segundero
230 NEXT t
```

Este reloj funcionará alrededor de 55.5 horas a causa de la línea 60, pero puede incrementar su duración fácilmente. Observe como la pulsación viene definida por la línea 210. Parece que debería emplearse PAUSE 50 (en lugar de 42), pero los 8 restantes es el tiempo que el computador toma para sus cálculos y hay que tenerlo en cuenta. La mejor manera es hacerlo por tanteo, comparando el reloj del computador con uno de verdad y ajustando la línea 210 para afinarlo (Es difícil hacerlo muy exacto : un ajuste de un cuadro cada segundo es un 2% y equivale a media hora por día).

Existe una manera mucho más exacta de medir el tiempo. Se logra empleando el contenido de ciertas posiciones de memoria. Los datos guardados se recuperan usando PEEK. El capítulo XXV explica esto con más detalle. La expresión usada es :

```
(65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50
```

Esto proporciona el número de segundos transcurrido desde que el computador ha sido conectado (alcanza a cerca de 3 días y 21 horas, que es cuando vuelve a cero).

Este programa de relój, aplica estos principios :

```
10 REM Dibujamos la esfera del reloj
20 FOR n=1 TO 12
30 PRINT AT 10-10*COS(n/6*PI),16+10*SIN(n/6*PI);n
```



```
40 NEXT n
50 DEF FN t()=INT  (65536*PEEK 23674+256*PEEK 23673+
   PEEK 23672)/50) REM segundos desde la conexión
100 REM Ahora ponemos en marcha el reloj
110 LET t1=FN t()
120 LET a=t1/30*PI:REM angulo del segundero en radianes
130 LET sx=72*SIN a:LET sy=72*COS a
140 PLOT 131,91:DRAW OVER 1;sx,sy:REM dibujar manecilla
150 LET t=FN t()
160 IF t<=t1 THEN GO TO 200:REM espera hasta siguiente
   pulsación
170 PLOT 131,91:DRAW OVER 1;sx,sy:REM borrar manecilla
180 LET t1=t:GO TO  20
```

El reloj interno que emplea este método tiene una precisión de cerca del 0.01% que representa unos 10 segundos de diferencia por día, pero se detiene el conteo cada vez que se emplea BEEP, u opera con el casette, o emplea la impresora o cualquier otro accesorio que emplee con el computador. Todo ello reporta una pérdida de tiempo.

Los números PEEK 23674, PEEK 23673 y PEEK 23672 se mantienen en el interior del computador y son usados para contar cincuentavos de segundo. Cada uno se halla entre 0 y 255 y se incrementan gradualmente a través de los números de 0 a 255; después de 255 vuelven directamente a 0.

El que se incrementa más a menudo es PEEK 23672. Cada 1/50 segundos se incrementa en 1. Cuando está a 255, el siguiente incremento lo pone a cero y aúmenta a PEEK 23673 en 1. Cuando (cada 255/50 seg.)PEEK 23673 pasa de 255 a cero incrementa , a su vez, a PEEK 23674 en 1. Esto sirve para dar una idea de como funciona la expresión anterior.

Ahora, veámolos detenidamente suponga que nuestros tres números valen cero (para PEEK 23674), 255 (para PEEK 23673) y 255 (para PEEK 23672). Esto significa que, al cabo de aprox. 21 minutos después de la puesta en marcha, nuestra expresión sería :

```
(65536*0+256*255+255)/50= 1310.7
```

Pero hay un peligro oculto. La próxima vez que cuente un 1/50 de segundo, los tres números cambiarán a 1, 0 y 0. Muy a menudo, esto sucederá mientras está evaluando la expresión :el computador valorará PEEK 23674 como 0, pero luego cambiará a los otros a cero antes de que pueda aplicarles la función PEEK. La respuesta entonces será

```
(65536*0+256*0+0)/50=0
```

totalmente desoladora.

Una regla simple para evitar este problema es evaluar la expresión por duplicado sucesivamente y coger la respuesta mayor.

Si observa atentamente el programa anterior verá que ya se ha tenido en cuenta esta regla. Hay un truco para aplicarla y es el siguiente.Define las funciones

```
10 DEF FN m(x,y)=(x+y+ABS(x-y)) /2:REM el mayor de x e y
20 DEF FN u()=(65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50:
   REM tiempo, puede estar equivocado
30 DEF FN t()=FN m(FN u(),FN u()):REM tiempo, correcto
```

Puede Vd. cambiar los tres números contadores de manera que nos den el tiempo real en lugar del tiempo transcurrido desde la conexión del computador. Por ejemplo, para ajustar la hora a las 10.00 de la mañana, observará que ello equivale a 10*60*60*50= 1800000 cincuentavos de segundo y que

```
1800000=65536*27+256*119+64
```

Para igualar los tres números a 27, 119 y 64, debe hacer

```
POKE 23674,27:POKE 23673,119:POKE 23672,64
```

Naturalmente, en paises cuya frecuencia de la red sea 60 ciclos, deben colocar esta cantidad en lugar de 50.

La función INKEYS$ (sin argumento) realiza la lectura del teclado. Si está pulsando en este momento una tecla (o SHIFT más otra tecla), entonces el resultado es el carácter que dicha tecla dá en modo L; en caso contrario el resultado es una cadena vacía.

Pruebe este programa que trabaja igual que una máquina de escribir.

```
10 IF INKEY$ <> "" THEN GO TO 10
20 IF INKEY$ = "" THEN GO TO 20
30 PRINT INKEY$;
40 GO TO 10
```

La línea 10 espera que retire sus dedos del teclado y la línea 20 que pulse una nueva tecla.

Recuerde que, al contrario que INPUT, INKEY$ no le espera ninguna introducción. Así pues no pulse ENTER, pero, por otro lado, si no entra nada tampoco el computador hará nada.

Ejercicios :

1.- ¿ Que sucede si omite la línea 10 del programa anterior ?.

2.- Otra forma de empleo de INKEY$ es combinándola con PAUSE, como en este programa alternativo del anterior :

```
10 PAUSE 0
20 PRINT INKEY$;
30 GO TO 10
```

Para efectuar este cometido ¿ porqué es fundamental que una pausa no se interrumpa si se encuentra pulsando todavía una tecla cuando aquella empieza ?

3.- Adapte el programa del reloj que hemos visto de manera que muestre, además las manecillas de las horas y minutos, dibujándolos cada minuto. Si se siente con ánimos, apáñeselas para que cada cuarto de hora se imprima o suceda algo (¿porqué no produce el sonido del BIG BEN por ejemplo?) mediante el BEEP. (Ver próximo capítulo).

4.- (Sólo para sádicos). Pruebe esto :

```
10 IF INKEY$="" THEN GO TO 10
20 PRINT AT 11,14;"SADICO¡"
30 IF INKEY$ <> "" THEN GO TO 30
40 PRINT AT 11,14;"    ":REM 4 espacios
50 GO TO 10
```


CAPITULO XIX .- BEEP

Resumen : BEEP

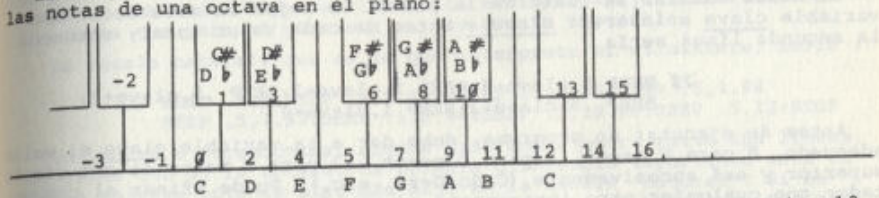
Si todavía no se ha enterado de que el ZX Spectrum tiene un altavoz incorporado, lea el manual de introducción antes de continuar..

El altavoz se pone en acción mediante el empleo de BEEP, en la forma,

BEEP duración, tono

donde, como ya es habitual,'duración' y 'tono' representan cualquier expresión numérica. La duración viene dada en segundos y el tono en semitonos por encima del 'do' central (C), empleando números negativos por debajo de dicha nota.

El diagrama siguiente muestra las equivalencias entre los tonos de las notas de una octava en el piano:



Para conseguir notas más altas o más bajas, debe sumar o restar 12 a cada octava que suba o baje.

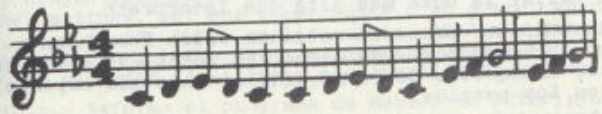
Si tiene un piano a su disposición cuando esté programando una melodía, posiblemente el diagrama anterior será suficiente para obtener los valores tonales. Si, por el contrario, está transcribiendo directamente desde una partitura, le sugerimos que dibuje un diagrama del pentagrama con los valores tonales escritos en cada línea y espacio, teniendo en cuenta la tecla.

Por ejemplo, entre :

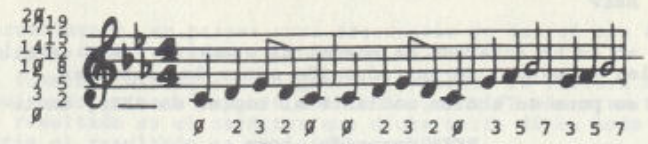
```
10 PRINT "Frere Gustav"
20 BEEP 1,0:BEEP 1,2:BEEP .5,3:BEEP .5,2:BEEP 1,0
30 BEEP 1,0:BEEP 1,2:BEEP .5,3:BEEP .5,2:BEEP 1,0
40 BEEP 1,3:BEEP 1,5:BEEP 2,7
45 BEEP 1,3:BEEP 1,5:BEEP 2,7
55 BEEP .75,7:BEEP .25,8:BEEP .5,7:BEEP .5,5:
   BEEP .5,3:BEEP .5,2:BEEP 1,0
65 BEEP .75,7:BEEP .25,8:BEEP .5,7:BEEP .5,5:
   BEEP .5,3:BEEP .5,2:BEEP 1,0
70 BEEP 1,0:BEEP 1,-5:BEEP 2,0
75 BEEP 1,0:BEEP 1,-5:BEEP 2,0
```

Cuando lo ejecute, debe obtener la marcha fúnebre de la primera sinfonía de Mahler, el fragmento donde los duendes entierran al soldado de caballería de los EEUU.

Suponga, por ejemplo, que su melodía está escrita en clave C menor, como el Mahler anterior. El principio empieza así :



pudiendo escribir los valores tonales de las notas de la siguiente manera :



Sólo hemos añadido dos líneas, para facilitar las medidas. Observe como el 'mi'bajo en la armadura, afecta no sólo al 'mi' del espacio superior, bemolizándolo del 16 al 15 sino que también el 'mi' de la línea del pié, ha sido bemolizado de 4 a 3. Debería ser bastante fácil ahora encontrar el tono de cualquier nota del pentagrama.

Si desea cambiar la clave de la pieza, lo mejor es establecer una variable clave e insertar clave + antes de cada valor tonal; entonces la segunda línea sería :

```
20 BEEP 1,clave+0:BEEP 1,clave+2:BEEP .5,clave+3:
   BEEP .5,clave+2:BEEP 1,clave+0
```

Antes de ejecutar un programa, debe dar a la variable clave el valor adecuado, 0 para C menor, 2 para D menor, 12 para C menor de una escala superior y así sucesivamente (C=do;D=re, etc,). Puede afinar el computador con cualquier otro instrumento ajustando clave y usando valores fraccionarios.

También puede definir la duración de cada nota. Dado que es una pieza relativamente lenta, hemos establecido un segundo para una negra y basado el resto en ello : medio segundo para una corchea, etc...

Es más flexible establecer una variable llamada negra para definir la longitud de una negra y especificar las duraciones con respecto a ella. Entonces la citada línea 20, se convertiría en :

```
20 BEEP negra,clave+0:BEEP negra,clave+2:BEEP negra/2,
   clave+3:BEEP negra/2,clave+2:BEEP negra,clave+0
```

(quizás decida dar a negra y clave, nombres más cortos y manejables).

Dándole valores apropiados a negra, puede variar fácilmente la velocidad de la pieza. Recuerde que al tener un sólo altavoz,el computador solamente puede interpretar una nota cada vez, por lo que no puede interpretar composiciones polifónicas. Si desea algo más, deberá cantar usted mismo.

Experimente programando melodías por su cuenta, empezando por tonadillas sencillas. Si no tiene piano ni partituras escritas, consiga una flauta o una armónica e interprete las notas que obtenga. Podría crear una tabla que diera el valor tonal de cada nota que tocara en el instrumento.

Entre :

```
FOR n=0 TO 1000:BEEP .5,n:NEXT n
```

tocará las notas en sentido ascendente hasta que se detenga con un informe B Integer out of range (B Entero fuera de escala). Puede hacerle imprimir n para hallar la nota más alta que interpretó.

Pruebe el mismo efecto pero con notas en orden descendente. Las muy bajas sonarán como chasquidos ; de hecho las notas más altas también están compuestas de chasquidos de forma similar pero más rápidos, de forma que el oído no los percibe.

Solamente las notas de la gama media, son adecuadas para la música.

Las demasiado bajas suenan como chasquidos y las muy altas se agudizan y tienden a distorsionar ligeramente.

Entre esta línea de programa :

```
10 BEEP .5,0:BEEP .5,2:BEEP .5,4:BEEP .5,5:BEEP .5,7:
   BEEP .5,9:BEEP .5,11:BEEP .5,12:STOP
```

(evidentemente, .5 equivale a 0.5)

Esta línea interpreta la escala en DO mayor, que emplea todas las notas blancas del piano desde la nota DO media hasta el DO inmediatamente superior. La forma en que se armoniza esta escala es igual que en un piano y se llama afinación 'plácida' a causa de que el intervalo de altura de un semitono es el mismo a lo largo de toda la escala. Un violinista , sin embargo, interpretaría la escala ligeramente diferente para hacer las notas más agradables al oído. Simplemente haciendo deslizar sus dedos imperceptiblemente arriba o abajo de las cuerdas en una forma que para el pianista es obviamente imposible.

La escala natural, que es lo que interpreta el violinista, sería :

```
'20 BEEP .5,0:BEEP .5,2.0039:BEEP .5,3.86:BEEP .5,4.98
    BEEP .5,7.02:BEEP .5,8.84:BEEP .5,10.88:BEEP .5,12:STOP
```

Vd. podrá o no,encontrar alguna diferencia entre estas dos líneas. Hay gente que sí la notará. La primera diferencia importante está en que la tercera nota es algo más baja en la escala 'natural'. Si es Vd. realmente un perfeccionista quizás le gustaría programar sus sintonías usando la escala natural en lugar de la plácida anterior. El inconveniente está en que, a pesar de que en clave de DO funciona perfectamente, con otras claves no es así (cada clave tiene su propia escala natural) y, en algunas, funciona verdaderamente fatal. La escala plácida está sólo ligeramente desplazada y trabaja igual de bien en todas las claves.

Esto, para el computador,no es ningún problema, por supuesto, ya que puede emplear el truco de añadir una variable clave.

Determinada música, especialmente la india, usa intervalos de altura inferior a un semitono. Puede programar esto con la sentencia BEEP sin ningún problema ; por ejemplo, el cuarto de tono por encima del DO medio tiene un valor tonal de 0.5.

Puede hacer que el teclado 'suene' cada vez que apriete una tecla, mediante

```
POKE 23609,255
```

el segundo número determina la longitud (duración) del pitido (pruebe diversos valores entre 0 y 255). Cuando es 0, la duración es tan corta que suena como un chasquido.

Si está interesado en profundizar en el sonido del ZX Spectrum, p.e. oírlo en otro lado que no sea el pequeño altavoz interno, observará que la señal de BEEP está presente en ambas conexiones de EAR y MIC, en la parte posterior del computador. Es ligeramente más alta en la salida EAR. Puede emplearla para conectar unos auriculares a su Spectrum. Ello no anulará el altavoz interno. Si necesita un sonido mayor, entonces puede conectar un amplificador (la salida MIC es la más adecuada en este caso) o también grabar el sonido en cinta y reproducirlo más adelante.

No hay ningún peligro de dañar al Spectrum, incluso aunque cortocircuite ambas salidas, así que experimente a su gusto. En determinados casettes, en posición PLAY, amplifican la señal que les entra por MIC procedente del EAR del Spectrum. No desespere si el suyo permanece mudo. Hay gran cantidad de sistemas para amplificar el sonido resultante.

Ejercicio.- Reforme el programa de Mahler de principio del capítulo, usando bucles FOR...NEXT para repetir las rayas del compás.

CAPITULO XX.- ALMACENAMIENTO EN CINTA

Resumen : LOAD, SAVE, VERIFY MERGE (Cargar, guardar,verificar y mezclar).

El método básico de empleo del casette para guardar (SAVE), cargar (LOAD) y verificar (VERIFY) programas se vió en el manual de introducción. Este capítulo debe leerse y comprobar varias veces los procedimientos descritos antes de seguir adelante.

Hemos visto como LOAD borra el programa que hubiera en el computador y sus variables antes de cargar el nuevo desde la cinta o casette; existe otra sentencia , MERGE (mezclar) que no actúa así. MERGE sólo borra una línea del antiguo programa o el programa entero si el nuevo programa a entrar tiene una línea del mismo número o si tiene el mismo nombre que el anterior.Entre el programa de la página 32 (cap XI) del juego de dados y guárdelo en cinta con el nombre "dados". A continuación entre lo siguiente !

```
1 PRINT 1
2 PRINT 2
10 PRINT 10
20 LET x=20
```

y proceda a su verificación pero en lugar de VERIFY"dados" escriba :

```
MERGE "dados"
```

Si procede a un listado del programa observará que las líneas 1 y 2 han permanecido intactas, pero las líneas 10 y 20 han sido sustituidas por las del programa de dados. x también ha permanecido (pruebe PRINT x).

Hemos visto hasta aquí formas sencillas de empleo de las cuatro sentencias usadas con el casette.

SAVE guarda(graba) el programa y sus variables en el casette.
VERIFY coteja el programa y las variables del casette con el programa y variables que todavía permaneten en el ordenador.
LOAD vacía al computador de todos sus programas y variables, reemplazándolos por nuevos procedentes del casette.
MERGE es igual a LOAD excepto en que no borra un antiguo programa o variable a menos que su número de línea o nombre sea el mismo que el nuevo procedente del casette.

En cada una, la sentencia es seguida por una cadena: en SAVE proporciona el nombre para guardar el programa en cinta, mientras que para las otras tres, dice al computador que nombre de programa debe buscar. Mientras está buscando, imprime el nombre de cada programa que va encontrando. Existen un par de singularidades que vamos a explicar.

A VERIFY, LOAD y MERGE puede seguirlas por una cadena vacía en lugar del nombre a buscar; entonces el computador se despreocupa del nombre y toma el primer programa que encuentra en su camino.

Una variante de SAVE toma la forma :

```
SAVE cadena LINE número
```

Un programa guardado empleando este sistema se graba de manera tal que cuando se recupera mediante LOAD (pero no MERGE), salta automáticamente a la línea con el número citado, ejecutándose a continuación partiendo de esta línea.

Hasta ahora, la única clase de información que hemos guardado en casette, han sido programas junto con sus variables. Hay también dos cosas distintas llamadas matrices y bytes.

Las matrices se tratan con ligeras diferencias :

Puede guardar matrices en cinta empleando DATA dentro de la sentencia SAVE en la forma

```
SAVE cadena DATA(nombre matriz) ()
```

cadena es el nombre que la información debe tener en la cinta y funciona de la misma manera que cuando guarda un programa.

El nombre de la matriz especifica la matriz que desea guardar, así que se trata de una letra sóla o seguida por $. Recuerde los paréntesis que siguen; seguramente los creerá innecesarios pero debe ponerlos para facilitar las cosas al ordenador.

No confunda los diferentes cometidos de cadena y nombre matriz. Si escribe, por ejemplo,

```
SAVE "Medida" DATA b()
```

entonces SAVE toma la matriz b del computador y la guarda en cinta bajo el nombre "Medida". Cuando entre

```
VERIFY 'Medida" DATA b()
```

el computador buscará un número de matriz guardado en cinta bajo el nombre "Medida" (cuando lo encuentre lo imprimirá 'Numero matriz:medida') y cotejará dicha mátriz con la denominada b del computador.

```
LOAD "Medida" DATA b()
```

encontrará la matriz en la cinta y luego,(si hay espacio para ella en el computador) borrará cualquier matriz que previamente hubiera con el nombre b, cargando a continuación la nueva matriz desde la cinta, llamándola b.

No se puede emplear MERGE con matrices almacenadas.
Puede guardar matrices de caracteres (o de cadenas) exactamente de la misma manera. Cuando el computador está buscando en la cinta y encuentra una de éstas, imprime 'Character array :(matriz de caracteres)seguido por el nombre correspondiente. Cuando está cargando una matriz, borrará, no sólo cualquier otra con el mismo nombre, sino también cualquier cadena que lo tenga igual.

El almacenamiento de bytes se emplea en partículas de información sin especificar para qué se emplea dicha información; puede tratarse de imágenes de televisión, gráficos predefinidos,o cualquier cosa hecha por uno mismo. Se emplea la palabra CODE, tal como

```
SAVE "dibujo" CODE 16384,6912
```

La unidad de almacenamiento en memoria el el byte (un número entre 0 y 255) y cada byte tiene una dirección (que es un número entre 0 y 65535). El primer número después de CODE, especifica la dirección del primer byte a guardar en cinta y el segundo, el número de bytes a guardar.. En nuestro caso, 16384 es la dirección del primer byte del archivo de pantalla (que contiene la pantalla entera de la televisión) y 6912 es el número de bytes empleados, así que estamos guardando una copia del dibujo en pantalla (pruébelo). El nombre "dibujo" opera igual que los nombres de los programas.
Para cargarlo de nuevo, emplear

```
LOAD "dibujo" CODE
```

Puede colocar números a continuación de CODE, en la forma

LOAD nombre CODE principio,longitud

Aquí, longitud es solamente una medida de seguridad; cuando el computador ha encontrado los bytes en la cinta con el nombre correcto, no los cargará si son más largos de lo previsto, dado que , obviamente, si duran más, podrían sobreimprimirse sobre algo no deseado. En este caso dará el informe de error R Tape loading error (R error al cargar).Si no especifica longitud determinada, entonces el computador cargará los bytes no importa cuántos sean..

Principio, indica la dirección en donde va a colocarse el primer byte que carguemos y que puede ser diferente de la que teníamos al guardar el programa. En caso de ser la misma, puede omitir principio en la sentencia LOAD.

CODE 16384,6912 es tan útil para guardar y cargar la imagen que puede reemplazarlo todo por SCREEN$. Por ejemplo,

```
SAVE "dibujo" SCREEN$
LOAD "dibujo" SCREEN$
```

Este es uno de los pocos casos en que VERIFY no actúa, dado que VERIFY imprime los nombres de lo que va encontrando en la cinta, así que cuando procedería a la verificación, el archivo de pantalla (la imagen) habría cambiado y fallaría la comprobación. En todos los demás casos debería emplear VERIFY siempre que use SAVE.

A continuación daremos una descripción completa de las cuatro sentencias contempladas en este capítulo.

Nombre se da a cualquier expresión de cadena y describe el nombre bajo el que la información se ha guardado en cinta. Debe estar formado por caracteres ASCII y sólo se emplearán los diez primeros que tenga dicho nombre (evidentemente, sólo en aquellos casos de nombres de más de diez caracteres)

Existen cuatro clases de información que pueden guardarse en cinta : programas y variables (juntos), matrices numéricas, matrices de caracteres y bytes sueltos.

Cuando VERIFY, LOAD y MERGE están buscando por la cinta, información con un determinado nombre o de una clase específica, imprimen en la pantalla la clase y nombre de la información que encuentran. La clase se indica con 'Program:', 'Number array:','Character array:' o 'Bytes:' (Respectivamente Programa:, Matriz numérica:,Matriz de caracteres:, o Bytes:) Si el nombre fuera una cadena vacía, tomarían el primer lote de información de la clase adecuada, prescindiendo del nombre.

SAVE.- (Guarda, almacena, salva, etc...)

Guarda información en cinta bajo un nombre dado. Se produce el error F cuando no existe nombre o éste tiene más de 10 caracteres.

SAVE siempre imprime un mensaje 'Start tape, then press any key (Ponga en marcha la cinta (en grabación naturalmente) y pulse cualquier tecla) y espera que se pulse una tecla para empezar el proceso.

1). Programa y variables :

```
SAVE nombre
```

es la forma normal.

```
SAVE nombre LINE número de línea
```

guarda el programa y las variables en tal forma que LOAD funciona automáticamente en la forma

```
GO TO numero de linea
```


'SAVE nombre LINE'equivale a SAVE nombre LINE 1 de forma que cuando se carga dicho programa, se ejecuta automáticamente a partir de su principio.

2) Bytes :

```
SAVE nombre CODE principio,longitud
```

guarda tantos bytes como indica longitud a partir de la dirección que señala principio.

```
SAVE nombre SCREEN$
```

equivale a

```
SAVE nombre CODE 16384,6912
```

y guarda la imagen en pantalla en aquel momento.

3) Matrices :

```
SAVE nombre DATA letra()
```

o

```
SAVE nombre DATA letra $()
```

guarda las matrices cuyos nombres sean letras o letras $ (no pueden tener ninguna relación con nombre)..

VERIFY.-(Verifica, comprueba, etc...).

Coteja la información de la cinta con la que todavía tiene en memoria el computador. El fallo en VERIFY dará el error'R Tape Loading error"(error de carga en la cinta, precedido por la letra R)

1) Programa y variables :

```
VERIFY nombre
```

2) Bytes

```
VERIFY nombre CODE principio, longitud
```

Si los bytes de 'nombre', tienen mayor longitud que la indicada, dará el error R. En caso contrario, cotejará estos con los bytes en memoria empezando por la dirección principio.

```
VERIFY nombre CODE principio
```

coteja los bytes 'nombre" con los de la memoria empezando por la dirección indicada por 'principio'.

```
VERIFY nombre CODE
```

coteja los bytes definidos por 'nombre' en la cinta con los que tiene en memoria empezando por la dirección en que fueron guardados en el casette.(por el primer byte, por supuesto).

```
VERIFY nombre SCREEN$
```

equivale a

```
VERIFY nombre CODE 16384,6912
```

que es casi seguro que dará código de error.

3) Matrices:

```
VERIFY nombre DATA letra()
```

o

```
VERIFY nombre DATA letra$()
```

coteja la matriz correspondiente a 'nombre' de la cinta con la definida por 'letra' ó 'letra$" de la memoria.

LOAD.- (cargar)

Carga nueva información desde la cinta, borrando la que hubiere en la memoria.

1) Programa y variables :

LOAD nombre

borra el antiguo programa y variables y carga un programa y variables nuevos llamado 'nombre' desde el casette; si el programa fue guardado empleando SAVE nombre LINE se ejecutará automáticamente después de finalizada la carga.

Se producirá un error '4 Out of memory' (4 fuera de memoria) si no existe suficiente espacio para el nuevo programa y sus variables. En este caso el programa antiguo y sus variables no quedan borrados.

2) Bytes :

LOAD nombre CODE principio,longitud

Si los bytes correspondientes a 'nombre' en la cinta son más numerosos que 'longitud' dá un error R. En caso contrario los carga en la memoria empezando en la dirección 'principio' y los sobrepone a los que hubiera antes en memoria.

LOAD nombre CODE principio

carga los bytes correspondientes a 'nombre' en la memoria borrando los que hubiera allí, empezando por la dirección 'principio'.

LOAD nombre CODE

carga los bytes de 'name' desde la cinta en la memoria empezando en la dirección misma en que fueron guardados en su día en la cinta, y anulando los bytes que pudiera haber con anterioridad ahí.

3) Matrices :

LOAD nombre DATA letra()

o

LOAD nombre DATA letra$()

borra cualquier matriz ya existente llamada letra o letra$ creando una nueva a partir de la guardada en cinta.
Si no hubiera espacio para la nueva matriz, tendríamos el informe de error'4 Out of memory' y la matriz antigua no se borraría.

MERGE.- (Combinar, mezclar, etc...)

Carga nueva información del casette sin borrar la información que estuviera en la memoria.

1) Programa y variables :

MERGE nombre

combina el programa 'nombre' con el que ya estuviera en memoria, sólo anulando las líneas y variables del programa antiguo que coincidan en número o valor con las del programa que entramos desde el casette, las cuales tienen preferencia.

En caso de que no haya suficiente espacio en la memoria para albergar los dos programas juntos, se producirá el informe '4 Out of memory'

2) Bytes :

Imposible

3) Matrices :

Imposible

Ejercicios :

1.- Hacer un casette en donde, el primer programa, cuando se haya cargado, imprima un menú (lista de otros programas en el casette), le pide que elija un programa y luego lo cargue automáticamente.

2.- Tome el programa de piezas de ajedrez del capítulo XIV y pulse NEW :dichos gráficos no se perderán. No obstante no se conservan si desconecta el computador; si quiere guardarlos, debe hacerlo en cinta, empleando LOAD junto con CODE. La manera más fácil es guardar los 21 gráficos predefinidos mediante

SAVE "ajedrez" CODE USR "a",21*8

seguido por

VERIFY "ajedrez" CODE

Este es el sistema de bytes que se empleó para guardar el dibujo. La dirección del primer byte a guardar es USR"a", la dirección del primero de los ocho bytes que determinan la estructura del primer gráfico predefinido, y el número de bytes a guardar es 21*8 (8 bytes para cada uno de los 21 gráficos).

Para recuperarlos puede emplear normalmente :

LOAD "ajedrez"CODE

No obstante si trata de cargarlo en un Spectrum con diferente cantidad de memoria RAM o si ha movido los gráficos definibles hacia una dirección distinta (lo habrá hecho deliberadamente empleando técnicas más sofisticadas), entonces extremar el cuidado y emplear :

LOAD "ajedrez" CODE USR "a"

la instrucción USR ayuda en el caso de que los gráficos deban ser cargados en una diferente dirección.

CAPITULO XXI.- LA IMPRESORA ZX

Resumen : LPRINT, LLIST, COPY

Nota .- Ninguna de estas sentencias es standart en BASIC, a pesar de que LPRINT se emplea en otros computadores.

Si tiene una impresora ZX, va a aprender algunas operaciones para trabajar con ella. Este capítulo revisará las sentencias BASIC necesarias para trabajar con la impresora.

Las dos primeras, LPRINT y LLIST son idénticas a PRINT y LIST con la única diferencia de que emplean la impresora en lugar de la pantalla. (La L del principio es un 'residuo histórico' de cuando el BASIC se inventó. Se empleaba entonces la máquina de escribir eléctrica en lugar de televisión, por lo que PRINT significaba realmente print (impresión). Si quería grandes cantidades de datos de salida, debía emplear una impresora muy rápida a la salida y LPRINT significaba 'Line Printer PRINT" (PRINT en impresora en línea).

Pruebe por ejemplo, este programa .

```
10 LPRINT "Este programa".'
20 LLIST
30 LPRINT'"imprime el set de caracteres."'
40 FOR n=32 TO 255
50 LPRINT CHR$ n;
60 NEXT n
```

La tercera sentencia, COPY, imprime en papel, una copia de lo que se encuentre en la pantalla. Por ejemplo, teclee LIST para obtener un listado en la pantalla del programa anterior, y entre

```
COPY
```

Obsérvese que COPY no funciona con uno de los listados que el computador genera automáticamente, porqué éstos se anulan cada vez que se obedece un comando. Debe,o bien emplear LIST ante todo, o emplear LLIST y prescindir de COPY.

Puede detener siempre que quiera la impresora cuando está trabajando con el empleo de BREAK (CAPS SHIFT +SPACE).

Si ejecuta estas sentencias con la impresora desconectada, perderá todos los resultados y el programa continuará con la siguiente instrucción.

Pruebe lo siguiente :

```
10 FOR n=31 TO 0 STEP -1
20 PRINT AT 31-n,n;CHR$(CODE "0"+n);
30 NEXT n
```

Verá una configuración de caracteres bajando diagonalmente desde la esquina superior derecha hasta el fondo de la pantalla y entonces el programa se detiene imprimiendo scroll?

Ahora cambie AT 31-n,n, de la línea 20 , por TAB n. El programa se comportará exactamente igual que antes.

Ahora cambie PRINT de la misma línea, por LPRINT. En esta ocasión no tendrá scroll?(que no actúa en la impresora) y la imagen continuará con las letras de la F a la O.

Ahora, cambie TAB n por AT 31-n,n empleando de nuevo LPRINT. Esta vez obtendrá sólo una sola línea de símbolos. La razón de la diferencia estriba en que la salida (output) de LPRINT no se imprime de forma directa sino que almacena en una memoria intermedia (buffer) una imagen de una línea de longitud que el computador mandará a la impresora cuando llegue a ella. La impresión tiene lugar en los siguientes casos:

a) Cuando el 'buffer' está lleno,
b) Después de una sentencia LPRINT que no termine en coma o punto y coma.
c) Cuando una coma, apóstrofe o partícula TAB requiere línea nueva, ó
d) al final del programa, si queda algo por imprimir.

El apartado c) explica el porqué nuestro programa con TAB actúa de la forma que lo hace. Como por AT, el número de línea es ignorado y la posición LPRINT (igual que la posición PRINT, pero en la impresora en lugar de en la pantalla) se cambia al número de columna. Una partícula AT jamás puede mandar una línea a la impresora.

Ejercicio .-

1.- Haga una copia impresa de una senoide (SIN) ejecutando el programa del capítulo 17 y utilizando COPY.

CAPITULO XXII.- OTRAS EXPANSIONES

Existen otros accesorios que puede Vd. conectar al Spectrum.

El ZX Microdrive es un dispositivo de almacenamiento masivo de datos de alta velocidad y es mucho más flexible en su funcionamiento que un casette. Operará, no solamente con SAVE, VERIFY, LOAD y MERGE, sino también con PRINT, LIST, INPUT e INKEY$.

La red exterior se emplea para interconectar varios Spectrums (hasta un máximo de 64) de manera que cada uno puede comunicarse con los demás, con la ventaja, entre otras, de que sólo precisa un Microdrive para servir a varios computadores.

El interface RS 232 es una conexión standart que le permite enlazar el Spectrum con teclados, impresoras, computadores y multitud de otras máquinas incluso aunqué no estén específicamente diseñadas para el Spectrum.

Utilizarán algunas instrucciones que están en el teclado, que no actúan sin dichos accesorios conectados. Estas son OPEN#, CLOSE#, MOVE, ERASE, CAT y FORMAT (Respectivamente, ABRIR, CERRAR, MOVER, BORRAR, CAT abreviación de CATalogo, y FORMATO).

CAPITULO XXIII.- IN y OUT

Resumen : IN, OUT (Entrada, Salida)

El procesador puede leer la memoria (al menos con la RAM) y escribir en ella, empleando PEEK y POKE. Al procesador, por su cuenta, le tiene sin cuidado si la memoria es ROM, RAM o incluso si no existe ; solamente sabe que existen 65536 direcciones de memoria y que puede leer un byte de cada una (incluso si no significa nada) y escribir un byte en cada una (incluso aunqué éste se pierda). En una forma completamente análoga existen 65536 elementos llamados puertas I/O (Puertas de entrada y salida). Estas puertas son utilizadas por el procesador para comunicarse con expansiones exteriores tales como el teclado o la impresora y pueden controlarse en BASIC mediante la función IN y la sentencia OUT.

La función IN es parecida a PEEK.

IN dirección

Tiene un argumento, la dirección de puerta, y su resultado es un byte leido en esta puerta.

La sentencia OUT es parecida a POKE

OUT dirección, valor

escribe el valor citado en la puerta cuya dirección se especifica. El modo en que se interpreta la dirección depende en gran parte del resto del computador; a menudo, varias direcciones distintas significan lo mismo. En el Spectrum , es mejor imaginar las direcciones en binario ya que los bits individuales tienden a trabajar por su cuenta. Hay 16 bits, a los que llamaremos (empleando A para 'dirección') :

A15,A14,A13,A12,.....,A2,A1,A0

Donde A0 es el primer bit, A1 el segundo bit (de peso 2), A2 el de peso 4 , etc.... Los Bits A0,A1,A2,A3 y A4 son los más importantes. Normalmente valen 1, pero si cualquiera de ellos vale 0 es para indicar al computador una operación específica. El computador no puede realizar más de una cosa a la vez, de manera que sólo uno de estos bits puede ser cero. Los Bits A5,A6 y A7 son ignorados, de forma que si Vd. es un mago de la electrónica puede aprovecharlos para sí mismo. Las mejores direcciones a emplear son aquellas que son múltiplos de 32 menos 1, de manera que A0...A4 son todos '1'. Los Bits A8,A9, etc..se emplean a veces para proporcionar información adicional.

El byte leido o escrito tiene 8 bits que son frecuentemente denominados D7,D6,.....D1,D0 (usando D de "dato').. He aqui una lista de las direcciones de puertas usadas.

Hay un conjunto de direcciones de entrada que leen el teclado y también la salida EAR.

El teclado se divide en 8 medias filas de 5 teclas cada una

```
IN 65278 'lee' la media fila de CAPS SHIFT a V
IN 65022 'lee' la media fila de A a G
IN 64510 'lee' la media fila de Q a T
IN 63486 'lee' la media fila de 1 a 5
IN 61438 'lee' la media fila de 0 a 6
IN 57342 'lee' la media fila de P a 7
IN 49150 'lee' la media fila de ENTER a H
IN 32766 'lee' la media fila de SPACE a B
```

(Estas direcciones son 254+256*(255-2↑n) siendo el valor de n de 0 a 7)

En el byte leído, los bits DØ a D4 representan las cinco teclas de la media fila indicada (DØ para la tecla exterior,D4 para la más cercana al centro). El bit vale Ø si se aprieta la tecla y 1 si no. D6 es el valor en la salida EAR.

La dirección de puerta 254 en salida(OUT) excita el altavoz (D4) y el conector MIC (D3), estableciendo , además, el color del contorno (D2, D1 y DØ).

La dirección de puerta 251 acciona la impresora, tanto en lectura como en escritura : en lectura establece si la impresora está en disposición de recibir información, en escritura envía puntos que serán impresos.

Las puertas de direcciones 254, 247 y 239 se emplean para las expansiones accesorias citadas en el capítulo anterior.

Ejecute el siguiente programa :

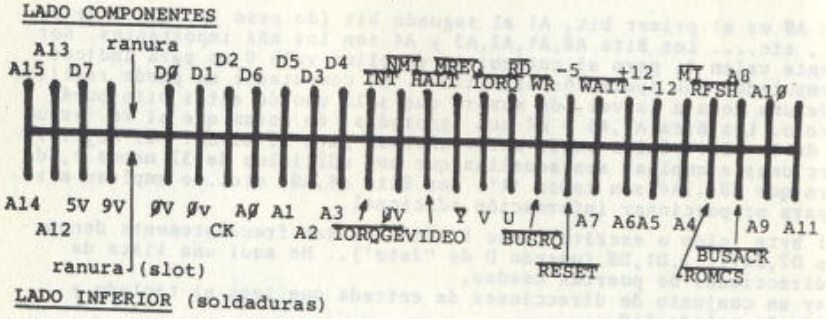
```
1Ø FOR n=Ø TO 7:REM número media fila
2Ø LET a=254+256*(255-2↑n)
3Ø PRINT AT Ø,Ø;IN a:GO TO 3Ø
```

y experimente pulsando teclas. Cuando se canse con las medias filas, apriete BREAK y, a continuación, entre

```
NEXT n
```

Los buses de control, datos y direcciones, están presentes en el conector de expansiones, en la parte trasera del Spectrum, de forma que puede realizar con el Spectrum la mayoría de las cosas que podría hacer con el Z 8Ø en solitario. En algunos casos, al intervenir el hardware del Spectrum, se le facilitarán incluso. Vea, a continuación, el diagrama de las citadas conexiones de expansión :

CONECTOR DE EXPANSIONES

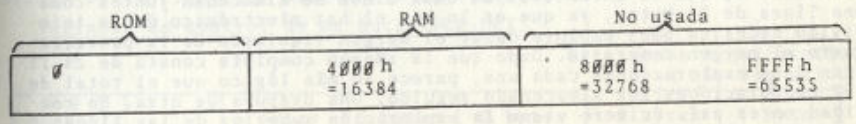


CAPITULO XXIV.- LA MEMORIA

Resumen : CLEAR

En las 'entrañas' del computador, toda la información se almacena en forma de bytes,p.e.los números comprendidos entre el Ø y el 255. Quizás pueda creer que ha guardado el precio de la lana o la dirección de su dentista así como suena; lo cierto es que todo ha sido convertido en largas colecciones de bytes (octetos) que es lo que realmente 've' el computador.

Cada posición donde puede almacenarse un byte posee una dirección, con un número entre Ø y FFFF (una dirección puede almacenarse como 2 bytes), así que puede imaginarse a la memoria como una larga fila de casillas numeradas, cada una de las cuales contiene un byte. No todas las casillas son iguales, sin embargo. En el modelo de 16 K., las casillas comprendidas entre el 8ØØØ y FFFF no existen. Las casillas comprendidas entre 4ØØØ y 7FFF son casillas RAM, lo que significa que puede alterar su contenido cuando lo desee y las comprendidas entre Ø y 3FFF son casillas ROM, que pueden 'verse' tranquilamente pero que no pueden modificarse. Todo lo que puede hacer es 'leer' todo lo que se puso ahí cuando se fabricó el computador.. Todos los números citados hasta ahora en este capítulo son, como ya habrá observado, en código hexadecimal (h).



Para inspeccionar el contenido de una casilla, emplearemos la función PEEK : su argumento será la dirección de la casilla y su resultado, el contenido de la misma. Por ejemplo, el siguiente programa imprime los primeros 21 bytes de la ROM (y sus direcciones).

```
1Ø PRINT"Direccion";TAB 8;"Byte"
2Ø FOR a=Ø TO 2Ø
3Ø PRINT a;TAB 8;PEEK a
4Ø NEXT a
```

Seguramente todos estos bytes no tendrán ningún significado para Vd. pero el computador (el microprocesador) los interpreta como instrucciones de lo que tiene que hacer.

Para cambiar el contenido de una casilla (en caso de memoria RAM), emplearemos la sentencia POKE. Tiene la forma

```
POKE direccion,nuevo contenido
```

en donde, 'direccion' y 'nuevo contenido' son expresiones numéricas. Por ejemplo, si entra

```
POKE 31ØØØ,57
```

significa que al byte cuya dirección es la 31ØØØ le ha dado un nuevo valor, en este caso el 57. Entre

```
PRINT PEEK 31ØØØ
```

Para comprobarlo (Pruebe la función POKE con otros valores, para comprobar que no es una simple casualidad). El nuevo valor a aplicar debe es-

tar comprendido entre -255 y +255, añadiéndosele 256 si es negativo.

La función POKE le proporciona un inmenso poder sobre el computador si sabe aplicarla y un inmenso poder destructor en caso contrario. Es muy sencillo, empleando POKE, colocar un número equivocado en una dirección incorrecta y echar a perder un largo programa que le llevó varias horas de trabajo. Afortunadamente, no le puede causar ningún daño al propio computador.

Vamos a echar a continuación un vistazo más detallado a la forma en que la memoria RAM es utilizada aunque si no le interesa el tema puede 'saltárselo' sin más problemas.

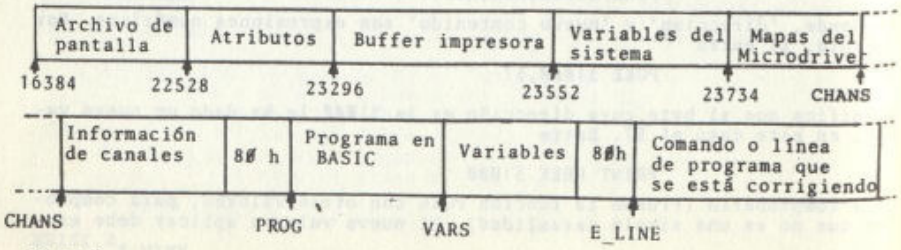
La memoria se divide en áreas diferentes (ver diagrama) para almacenar diferentes clases de información. Las áreas son dimensionadas solamente para la información que contienen habitualmente y, si se inserta información adicional en un punto dado (por ejemplo añadiendo una línea de programa o una variable) se logra espacio empujando hacia arriba a partir de este punto. Por el contrario, si se elimina información , todo se empuja hacia abajo.

El archivo de pantalla (display file) almacena la imagen en pantalla en aquel momento. Lo hace de forma bastante curiosa, de manera que es mejor no hacer ni PEEK ni POKE en dicho archivo. Cada posición de cada carácter en la pantalla está formada por 8x8 puntos y cada uno de dichos puntos puede ser 0 (fondo) ó 1 (texto). Empleando notación binaria podemos almacenar esta configuración como 8 bytes, uno por cada fila. No obstante, estos 8 bytes no se almacenan seguidos. Las correspondientes filas de los 32 caracteres de cada línea se almacenan juntos como una línea de 32 bytes, ya que es lo que el haz electrónico de la televisión necesita pues explora desde el margen izquierdo de la pantalla hasta el margen contrario. Dado que la imagen completa consta de 24 líneas de 8 exploraciones cada una, parece lo más lógico que el total de 192 exploraciones sea almacenado seguido, una después de otra. En realidad no es así. Primero viene la exploración superior de las líneas 0 a 7, a continuación la segunda exploración de estas mismas líneas, luego la tercera y así hasta la última (la octava) de estas líneas ; a continuación se repite lo mismo para las líneas 8 a 15 y luego, para las líneas 16 a 23. La conclusión final de todo ello es que si está habituado a un computador que emplea PEEK y POKE en la pantalla, tendrá que sustituirlos por SCREEN$ y PRINT AT , o por PLOT y POINT.

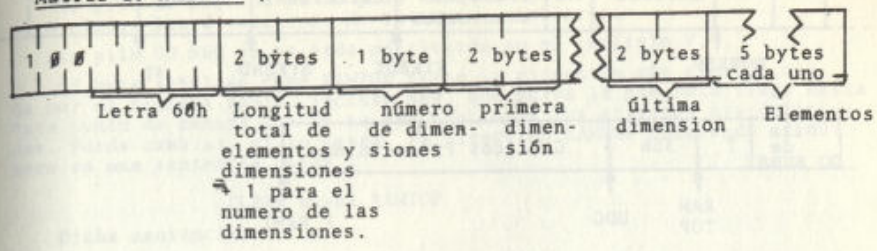
Los atributos son los colores y demás de cada posición de carácter, empleando el formato de ATTR. Estos se hallan almacenados línea por línea como era de esperar.

El 'buffer' de la impresora almacena los caracteres destinados a ésta.

Las variables del sistema contienen varias piezas de información para indicar al computador en que estado se encuentra. Hay una lista completa en el próximo capítulo pero, de momento, nótese que hay algunas (llamadas CHANS,PROG,VARS,E_LINE, etc) que contienen las direcciones de los límites entre las distintas áreas de la memoria. No son variables de BASIC y sus nombres no significan nada para el computador.



Matriz de números ;



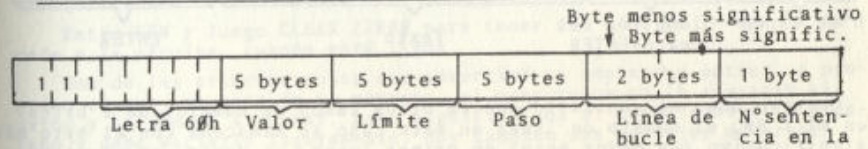
El orden de los elementos es el siguiente :

primero, los elementos cuyo primer subíndice es 1, luego los elementos cuyo primer subíndice es 2, a continuación los que tienen el 3 de primer subíndice y así sucesivamente para todos los valores posibles del primer subíndice.

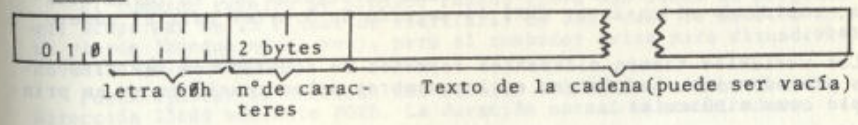
Los elementos con un primer subíndice dado, se ordenan de la misma forma empleando el segundo subíndice y así sucesivamente hasta el último.

Como ejemplo, los elementos de la matriz c de 3*6 del capítulo XII, se almacenan en el orden b(1,1) b(1,2) b(1,3) b(1,4) b(1,5) b(1,6) b(2,1) b(2,2)...b(2,6) b(3,1) b(3,2)...b(3,6).

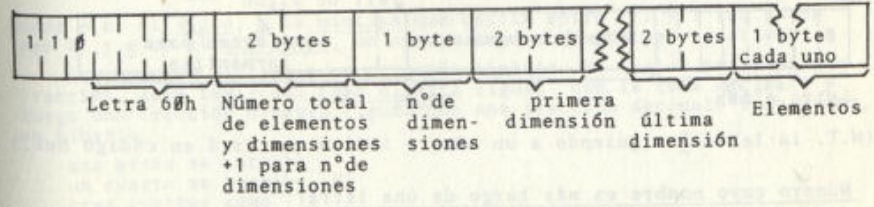
Variable de control de un bucle FOR-NEXT :



Cadena :

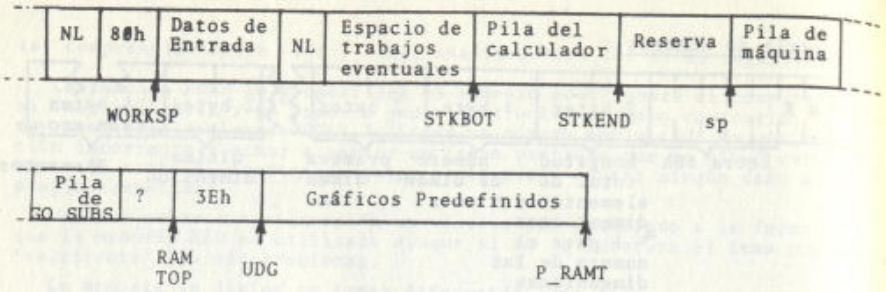


Matriz de caracteres :



El calculador es la parte del sistema en BASIC que trata con la aritmética y la mayoría de los números con qué opera están en la pila del calculador (calculator stack).

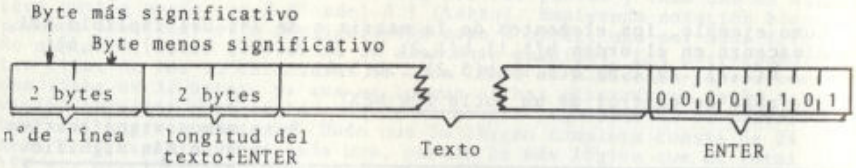
La parte de reserva contiene el espacio libre hasta el momento.



Los mapas del Microdrive se emplean solamente con éste. Normalmente no existe nada ahí.

La información de canales contiene información sobre los dispositivos de entrada y salida, es decir, el teclado (con la mitad inferior de la pantalla), la parte superior de la pantalla y la impresora.

Cada línea de programa en BASIC adopta la forma :

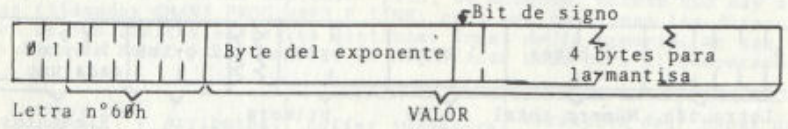


Observe que, al contrario que en otros casos de números de 2 bytes en el Z 80, el número de línea en este caso se almacena con el byte más significativo (de mayor peso) en primer lugar, es decir, en el mismo orden en que normalmente se escriben.

Una constante numérica en el programa es seguida por su forma binaria, empleando el carácter CHR$ 14 seguido de 5 bytes para el número citado.

Las variables tienen diferentes formatos de acuerdo con sus diferentes procedencias. Las letras en los nombres deben considerarse en principio como minúsculas.

Número cuyo nombre es una única letra :



(N.T. la letra h siguiendo a un número indica que está en código hex.)

Número cuyo nombre es más largo de una letra :

La pila de máquina es la pila empleada por el microprocesador Z80 para alojar las direcciones de retorno y similares.

La pila GO SUB ya ha sido mencionada en el capítulo V.

El byte señalado como RAMTOP tiene la dirección más elevada usada por el sistema BASIC. Incluso NEW, que borra la RAM sólo llega hasta este punto de manera que no se ven afectados los gráficos pre-definidos. Puede cambiar, si lo desea, la dirección RAMTOP colocando un número en una sentencia CLEAR :

CLEAR nueva RAMTOP

Dicha sentencia :

a) borra todas las variables
b) borra el archivo de pantalla (igual que CLS)
c) coloca de nuevo (reset) la posición PLOT a la esquina inferior izquierda
d) Ejecuta una sentencia RESTORE
e) borra la pila GO SUB y la coloca en el nuevo RAMTOP, suponiendo que éste se encuentre entre la pila del calculador y el final real de la RAM. (En caso contrario deja el RAMTOP donde estaba).

La sentencia RUN siempre realiza CLEAR a pesar de que no cambia el RAMTOP.

Usando CLEAR de esta manera puede hacer subir el RAMTOP para lograr más espacio para el BASIC superponiendose a los gráficos predefinidos o bien, puede hacerlo bajar, para tener más RAM 'inmune' a los efectos de NEW.

Entre NEW y luego CLEAR 23800 para tener una idea de lo que le sucede a la máquina cuando está llena.

Una de las primeras cosas que advertirá si empieza a entrar un programa es que, al cabo de un momento, el computador no le aceptará más datos y se pondrá a zumbar. Esto significa que el computador está atestado y debe vaciarlo ligeramente. Hay también dos mensajes de error con un significado muy parecido, 4 Memory full (4 Memoria llena) y G No room for line (G No hay espacio para más líneas).

El zumbido también se produce cuando entra una línea de programa que ocupa más de 23 líneas de escritura. En este caso lo escrito no se pierde (aunque no se vea), pero el zumbador avisa para disuadirle de entrar algo más.

Puede ajustar la duración del zumbido colocando un número en la dirección 23608 mediante POKE. La duración normal tiene el número 64.

Cualquier número (a excepción de 0) debe escribirse como

$$\pm m \times 2^{e}$$

donde ± es el signo, m la mantisa que oscila entre 1/2 y 1 (no puede ser 1) y e es el exponente, un número entero (puede ser negativo).

Suponga que escribe m en notación binaria. Ya que se trata de una fracción, debe tener una coma binaria (igual que la coma decimal) y luego una fracción binaria (igual que una fracción decimal) : así pues, en binario,

una mitad se escribe .1
un cuarto se escribe .01
tres cuartos como .11
una décima como .0001100110011001100110011. ..etc... Con nuestro número m, al ser menor de 1, no tenemos bits antes del punto binario y al ser mayor de 1/2, el bit que sigue al punto decimal siempre será 1.

Para almacenar el número en el computador, emplearemos cinco by-

tes, tal como sigue :

a) se escriben los primeros ocho bits de la mantisa en el segundo byte (ya sabemos que el primero es 1), los siguientes 8 bits en el tercer byte, los siguientes en el cuarto y los últimos 8 bits en el quinto.
b) se reemplaza el primer bit del segundo byte (que hemos dicho es 1) por el signo, Ø para positivo y 1 para negativo.
c) se escribe el exponente +128 en el primer byte. Por ejemplo, supongamos que nuestro número es 1/1Ø

$$1/10=4/5x2^{-3}$$

Entonces la mantisa m es .11ØØ11ØØ11ØØ11ØØ11ØØ11ØØ11ØØ11ØØ en binario (dado que el 33 bit es 1, redondearemos el 32 de Ø a 1) y el exponente e es -3. Aplicando nuestras tres reglas obtenemos los cinco bytes.

este Ø indica signo +

ØØ111 11Ø1 ØØ1ØØ 11ØØ 11ØØ 11ØØ 11ØØ 11ØØ 11ØØ 11Ø1

-3+128 mantisa 4/5 excepto que el primer bit debería ser 1 para el exponente

Existe una manera alternativa de almacenar números enteros entre -65535 y +65535 :

a) El primer byte es Ø
b) El segundo byte es Ø para un número positivo y FFh para uno negativo.
c) Los bytes tercero y cuarto son el menos y el más significativo (de menor y mayor peso respectivamente) del número (o el número +131072 si es negativo).
d) El quinto byte es Ø.

CAPITULO XXV.- LAS VARIABLES DEL SISTEMA

Los bytes de memoria entre 23552 y 23733 se reservan para usos específicos por el sistema. Pueden analizarse con PEEK si desea averiguar algunas cosas del sistema y algunas de ellas pueden alterarse mediante POKE de forma bastante útil. Más adelante se detallarán con sus respectivas aplicaciones.

Estas son las llamadas variables del sistema y tienen nombres concretos que no deben confundirse con las variables usadas en BASIC. El computador ignorará estos nombres de variables, por lo que solamente se dan como 'mnemonics'(reglas mnemotécnicas) para los humanos.

Las abreviaciones de la columna 1 tienen los siguientes significados :

X Las variables no deben ser modificadas por POKE ya que el sistema podría quebrantarse.
N Variando estas variables con POKE no causará ningún quebranto en absoluto.

El número de la columna 1 es el número de bytes de la variable. En el caso de 2 bytes el primero es el de menor peso (al contrario de lo que podría pensarse). Así pues, para colocar un valor v en una variable de dos bytes en la dirección n, utilice

```
POKE n,v-256*INT(v/256)
POKE n+1,INT (v/256)
```

y para examinar su valor (PEEK), emplee la expresión

```
PEEK n+256*PEEK(n+1)
```

Notas	Direccion	Nombre	Contenido
N8	23552	KSTATE	Se usa para leer el teclado
N1	2356Ø	LAST K	Mantiene el valor de la última tecla pulsada.
1	23561	REPDEL	Tiempo(en 1/5Ø de segundo) que debe mantenerse pulsada una tecla para que repita. Esto sucede con valor 35 pero puede probar otros valores.
1	23562	REPPER	Lapso (en 1/5Øs de segundo) entre sucesivas repeticiones de una tecla apretada(inicialmente 5).
N2	23563	DEFADD	Dirección de argumentos de gráficos predefinidos si se evalúa alguna. En caso contrario su valor es Ø.
N1	23565	K DATA	Almacena el segundo byte de control de colores procedente del teclado.
N2	23566	TVDATA	Almacena bytes de color, controles AT y TAB que van a la televisión.
X38	23568	STRMS	Direcciones de los canales de entrada y salida.
2	236Ø6	CHARS	Dirección del set de caracteres menos 256 (que comienza con un espacio y lleva el símbolo de copyright). Normalmente se halla en la ROM pero puede crear

Notas	Dirección	Nombre	Contenido
			un equivalente en la RAM y hacer que CHARS se refiera a él.
1	23608	RASP	Duración del zumbido de alerta.
1	23609	PIP	Duración del 'bip' del teclado.
1	23610	ERR NR	Código de informes menos 1. Empieza en 255 por lo tanto PEEK 23610 dá 255
X1	23611	FLAGS	Varios indicadores para controlar el sistema BASIC.
X1	23612	TV FLAG	Indicadores relacionados con la televisión.
X2	23613	ERR SP	Dirección de partícula en pila de máquina que se emplea como'error en retorno'.
N2	23615	LIST SP	Dirección de la posición de retorno de un listado automático.
N1	23617	MODE	Especifica el tipo de cursor (K,L,C,E ó G).
2	23618	NEWPPC	Línea a la que hay que saltar.
1	23620	NSPPC	Número de sentencia en una línea a la que hay que saltar. Haciendo POKE a NEWPPC y luego a NSPPC fuerza a saltar a una sentencia concreta dentro de una línea.
2	23621	PPC	Número de línea de la sentencia ejecutada en este momento
1	23624	BORDCR	Color del contorno*8; también contiene los atributos normalmente empleados en la mitad inferior de la pantalla.
2	23625	E PPC	Número de la línea en curso (con cursor del programa).
X2	23627	VARS	Dirección de variables.
N2	23629	DEST	Dirección de variable asignada.
X2	23631	CHANS	Dirección de los canales de datos.
X2	23633	CURCHL	Dirección de la información empleada en este momento como entrada y salida.
X2	23635	PROG	Dirección del programa BASIC
X2	23637	NXTLIN	Dirección de la siguiente línea del programa.
X2	23639	DATADD	Dirección de terminación de lá última partícula DATA.
X2	23641	E LINE	Dirección del comando que está siendo pulsado.
2	23643	K CUR	Dirección del cursor.
X2	23645	CH ADD	Dirección del siguiente carácter a interpretar : el carácter después del argumento de PEEK, o el NEWLINE al final de una sentencia POKE.
2	23647	X PTR	Dirección del carácter que sigue al signo
X2	23649	WORKSP	Dirección del espacio temporal de trabajo.
X2	23651	STKBOT	Dirección del final de la pila del calculador.
X2	23653	STKEND	Dirección del principio del espacio de reserva.
N1	23655	BREG	Registro b del calculador.
N2	23656	MEM	Dirección del área usada para la memoria del calculador (normalmente, aunque no siempre, llamada MEMBOT).
1	23658	FLAGS2	Más indicadores.
X1	23659	DF SZ	Número de líneas, incluyendo una en blanco, de la parte inferior de la pantalla.
2	23660	S TOP	Número de la línea superior de un programa en listados automáticos.
2	23662	OLDPPC	N°de línea al cual salta CONTINUE.
1	23664	OSPCC	Número dentro de la línea de la sentencia a la cual salta CONTINUE.
1	23623	SUBPPC	Número dentro de la línea que se está ejecutando

Notas	Dirección	Nombre	Contenido
N1	23665	FLAGX	Varios indicadores.
N2	23666	STRLEN	Longitud del destino tipo cadena asignado.
N2	23668	T ADDR	Dirección del tefmino siguiente en la tabla sintáctica (de muy dudosa utilidad).
2	23670	SEED	El origen de RND. Es la variable que se determina mediante RANDOMIZE.
3	23672	FRAMES	3 bytes(el de menor peso primero)del contador de cuadros. Se incrementa cada 20 ms. Ver capítulo XVIII.
2	23675	UDG	Dirección del primer gráfico predefinido. Puede cambiarlo, por ejemplo, en el caso de querer ahorrar espacio limitando el número de dichos gráficos.
1	23677	COORDS	x-coordenada del último punto trazado.
1	23678	" "	y-coordenada del último punto trazado.
1	23679	P POSN	número de la 33ª columna de posición de la impresora.
1	23680	PR CC	Byte de menos peso de la dirección de la siguiente posición para LPRINT (en el buffer de la impresora).
1	23681		No se utiliza.
2	23682	ECHO E	Número de la 33ª columna y 24ª línea (en la mitad inferior) del final de la entrada del buffer.
2	23684	DF CC	Dirección de la posición PRINT en el archivo de pantalla.
2	23686	DFCCL	Igual que DF CC para la parte inferior de la pantalla.
X1	23688	S POSN	Número de la 33ª columna para posición de PRINT.
X1	23689		Número de la 24ª fila para posición PRINT.
X2	23690	SPOSNL	Igual que S POSN para la mitad inferior.
1	23692	SCR CT	Cuenta los 'scrolls': siempre equivale al número de scrolls(desplazamientos) que se efectuarán antes de pararse con scroll? más la unidad (+1). Si altera este número mediante POKE con un número mayor que 1 (p. e. 255), la pantalla se desplazará una y otra vez sin preguntarle nada.
1	23693	ATTR P	Colores fijos en curso, etc (tal y como quedaron fijados por las sentencias de color.
1	23694	MASK P	Empleado para colores transparentes, etc. Cualquier bit que valga 1 indica que el bit del atributo correspondiente no se ha tomado de ATTR P sino de lo que ya había en la pantalla.
N1	23695	ATTR T	Colores provisionales en curso, etc.(tal como los fijaron las partículas de color).
N1	23696	MASK T	Igual que MASK P, pero temporal.
1	23697	P FLAG	Más indicadores.
N30	23698	MEMBOT	Area de memoria del calculador ; empleada para almacenar números que no pueden colocarse convenientemente en la pila (stack) del calculador.
2	23728		Sin uso.
2	23730	RAMTOP	Dirección del último byte del área de BASIC.
2	23732	P-RAMT	Dirección real del último byte de la RAM.

Este programa le mostrará los primeros 22 bytes del área de variables :

```
10 FOR n=0 TO 21
20 PRINT PEEK (PEEK 23627+256*PEEK 23628+n)
30 NEXT n
```

Trate de comparar la variable de control n con la tabla anterior. Ahora, sustituya la línea 20 por

```
20 PRINT PEEK (23755+n)
```

Esto le dará los primeros 22 bytes del área de programa. Compárelos con el propio programa.

CAPITULO XXVI.- UTILIZANDO CODIGO MAQUINA

Resumen : USR con argumento numérico.

Este capítulo se ha escrito pensando en aquellos que conocen el código máquina del Z80, el conjunto de instrucciones que el microprocesador Z80 emplea. Si Vd. no lo conoce pero le interesaría, existe una serie completa de libros al respecto. Debe buscar algo así como 'Código máquina para el Z80 para principiantes' o similar, de venta en cualquier librería especializada.Si encima, menciona al Spectrum, mejor que mejor.

Estos programas son escritos normalmente en lenguaje ensamblador, el cual, aunque parece criptografía, no es difícil de entender con un poco de práctica. (Puede ver las instrucciones del lenguaje ensamblador en el Apéndice A). No obstante, para ajecutarlos en el computador necesita codificar el programa en una secuencia de bytes,(de ahí el nombre de código máquina). Dicha traducción la efectúa normalmente el computador por su cuenta, usando un programa llamado ensamblador (assembler).. No existe ensamblador incorporado en el Spectrum pero puede adquirirlo en casette. En caso contrario, tendrá que hacer la traducción por sí mismo , cosa que resulta muy penosa si el programa es largo.

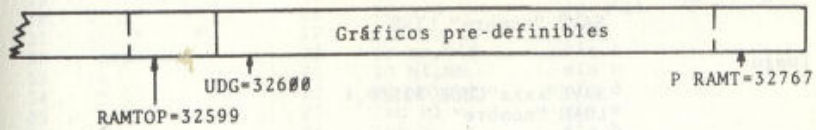
Veamos un ejemplo. El programa

```
ld bc,99
ret
```

carga el par de registros bc con 99.Esto se traduce en los 4 bytes de código máquina : 1,99,0 (para ld,bc,99) y 201 (para ret).(Si busca 1 y 201 en el Apéndice A, encontrará ld,bc,NN (siendo NN cualquier número de 2 bytes) y ret.)

Cuando haya eleborado su propio programa en código máquina, el siguiente paso es introducirlo en el computador. (Un ensamblador seguramente lo haría de forma automática). Debe decidir en que parte de la memoria desea colocarlo y es recomendable el conseguir espacio adicional entre el área del BASIC y los gráficos predefinidos.

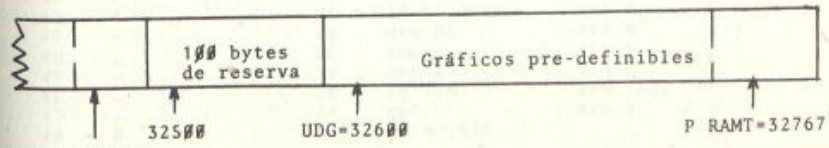
Suponga, por ejemplo, que tiene un Spectrum de 16 K. En este caso el límite superior de la RAM tiene :



Si entra

```
CLEAR 32499
```

esto le proporcionará un espacio de 100 (para asegurarse) bytes empezando en la dirección 32500.



Para introducir el programa en código máquina, debería ejecutar un programa en BASIC parecido a éste :

```
10 LET a=32500
20 READ n:POKE a,n
30 LET a=a+1:GO TO 20
40 DATA 1,99,0,201
```

(Esto se detendrá con el informe E Out of DATA cuando ha entrado ya los cuatro bytes indicados).

Para ejecutar el código máquina, se emplea la función USR, pero esta vez con argumento numérico, la dirección de partida. Su resultado es el valor del registro bc, retornando del programa en código máquina, así que si hace

```
PRINT USR 32500
```

obtendrá la respuesta 99.

La dirección de retorno al BASIC se realiza en la manera habitual, de forma que dicho retorno se logra con una instrucción ret del Z80. No debe emplear los registros iy e i en una rutina de código máquina.

Los buses de control, datos y direcciones se hallan al alcance en la parte trasera del Spectrum así que puede hacer con él lo mismo que haría con un Z80. No obstante, a veces, el hardware del Spectrum podría plantear algún problema.

Vea un diagrama del conector de expansiones del ZX Spectrum al final del capítulo XXIII cuando se habló de IN y OUT.

Puede guardar su programa en código máquina fácilmente con

```
SAVE "nombre" CODE 32500,4
```

Según parece no es posible guardarlo de forma que al cargarlo se ejecute automáticamente. Para lograrlo puede emplear un programa en BASIC.

```
10 LOAD ""CODE 32500,4
20 PRINT USR 32500
```

Haga primero

```
SAVE "nombre" LINE
```

y luego

```
SAVE"xxxx"CODE 32500,4
LOAD "nombre"
```

que cargará y ejecutará automáticamente el programa en BASIC, el cual a su vez, cargará y ejecutará el código máquina.

APENDICE A.- SET DE CARACTERES

Este es el set (conjunto) completo de caracteres del Zx Spectrum, con sus códigos decimales y hex. Si imagina los códigos como instrucciones en código máquina del Z 80, entonces, las columnas de la derecha proporcionan los 'nemónicos' del correspondiente lenguaje ensamblador. Como ya probablemente sabrá, si conoce este sistema, determinadas instrucciones del Z 80 son compuestos que empiezan con CB ó con ED ; Las dos columnas de la derecha muestran la relación de las mismas .

CODIGO	CARACTER	HEX	ENSAMBLADOR DEL Z80	CB +...	ED +...
0	no se usa	00	nop	rlc b	
1	" "	01	ld bc,NN	rlc c	
2	" "	02	ld (bc),a	rlc d	
3	" "	03	inc bc	rlc e	
4	" "	04	inc b	rlc h	
5	" "	05	dec b	rlc l	
6	PRINT coma	06	ld b,N	rlc (hl)	
7	EDIT	07	rlca	rlc a	
8	cursor izq.	08	ex af,af'	rrc b	
9	cursor derecha	09	add hl,bc	rrc c	
10	cursor abajo	0A	ld a,(bc)	rrc d	
11	cursor arriba	0B	dec bc	rrc e	
12	DELETE (Borrado)	0C	inc c	rrc h	
13	ENTER	0D	dec c	rrc l	
14	numero	0E	ld c,N	rrc (hl)	
15	no se usa	0F	rrca	rrc a	
16	control INK	10	djnz DIS	rl b	
17	control PAPER	11	ld de,NN	rl c	
18	control FLASH	12	ld(de),a	rl d	
19	control BRILLO	13	inc de	rl e	
20	control INVERSE	14	inc d	rl h	
21	control OVER	15	dec d	rl l	
22	control AT	16	ld d,N	rl (hl)	
23	control TAB	17	rla	rl a	
24	no se usa	18	jr DIS	rr b	
25	" "	19	add hl,de	rr c	
26	" "	1A	ld a,(de)	rr d	
27	" "	1B	dec de	rr e	
28	" "	1C	inc e	rr h	
29	" "	1D	dec e	rr l	
30	" "	1E	ld e,N	rr (hl)	
31	" "	1F	rra	rr a	
32	espacio	20	jr nz,DIS	sla b	
33	!	21	ld hl,NN	sla c	
34	"	22	ld(NN),hl	sla d	
35	#	23	inc hl	sla e	
36	$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	ld h,N	sla (hl)	
39	'	27	daa	sla a	
40	(	28	jr z,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	ld hl,(NN)	sra d	
43	+	2B	dec hl	sra e	
44	,	2C	inc l	sra h	
45	-	2D	dec l	sra l	
46	.	2E	ld l,N	sra (hl)	
47	/	2F	cpl	sra a	
48	0	30	jr nc,DIS		

CODIGO	CARACTER	HEX	ENSAMBLADOR DEL Z80	CB +...	ED +...
49	1	31	ld sp,NN		
5Ø	2	32	ld(NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
6Ø	<	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	>	3E	lda,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	4Ø	ld b,b	bit Ø,b	in b,(c)
65	A	41	ld b,c	bit Ø,c	out (c),b
66	B	42	ld b,d	bit Ø,d	sbc hl,bc
67	C	43	ld b,e	bit Ø,e	ld(NN),bc
68	D	44	ld b,h	bit Ø,h	neg
69	E	45	ld b,l	bit Ø,l	retn
7Ø	F	46	ld b,(hl)	bit Ø,(hl)	im Ø
71	G	47	ld b,a	bit Ø,a	ld i,a
72	H	48	ld c,b	bit 1,b	in c,(c)
73	I	49	ld c,c	bit 1,c	out(c),c
74	J	4A	ld c,d	bit 1,d	adc hl,bc
75	K	4B	ld c,e	bit 1,e	ld bc,(NN)
76	L	4C	ld c,h	bit 1,h	
77	M	4D	ld c,l	bit 1,l	reti
78	N	4E	ld c,(hl)	bit 1,(hl)	
79	O	4F	ld c,a	bit 1,a	ld r,a
8Ø	P	5Ø	ld d,b	bit 2,b	in d,(c)
81	Q	51	ld d,c	bit 2,c	out (c),d
82	R	52	ld d,d	bit 2,d	sbc hl,de
83	S	53	ld d,e	bit 2,e	ld(NN),de
84	T	54	ld d,h	bit 2,h	
85	U	55	ld d,l	bit 2,l	
86	V	56	ld d,(hl)	bit 2,(hl)	im 1
87	W	57	ld d,a	bit 2,a	ld a,i
88	X	58	ld e,b	bit 3,b	in e,(c)
89	Y	59	ld e,c	bit 3,c	out (c),e
9Ø	Z	5A	ld e,d	bit 3,d	adc hl,de
91	[	5B	ld e,e	bit 3,e	ld de,(NN)
92	\	5C	ld e,h	bit 3,h	
93	]	5D	ld e,l	bit 3,l	
94	↑	5E	ld e,(hl)	bit 3,(hl)	im 2
95		5F	ld e,a	bit 3,a	ld a,r
96	£	6Ø	ld h,b	bit 4,b	in h,(c)
97	a	61	ld h,c	bit 4,c	out (c),h
98	b	62	ld h,d	bit 4,d	sbc hl,hl
99	c	63	ld h,e	bit 4,e	ld(NN),hl
1ØØ	d	64	ld h,h	bit 4,h	
1Ø1	e	65	ld h,l	bit 4,l	
1Ø2	f	66	ld h,(hl)	bit 4,(hl)	
1Ø3	g	67	ld h,a	bit 4,a	rrd
1Ø4	h	68	ld l,b	bit 5,b	in l,(c)
1Ø5	i	69	ld l,c	bit 5,c	out (c),l
1Ø6	j	6A	ld l,d	bit 5,d	adc hl,hl
1Ø7	k	6B	ld l,e	bit 5,e	ld hl,(NN)
1Ø8	l	6C	ld l,h	bit 5,h	

CODIGO	CARACTER	HEX	ENSAMBLADOR DEL Z8Ø	CB +...	ED +...
1Ø9	m	6D	ld l,l	bit 5,l	
11Ø	n	6E	ld l,(hl)	bit 5,(hl)	
111	o	6F	ld l,a	bit 5,a	rld
112	p	7Ø	ld (hl),b	bit 6,b	in f,(c)
113	q	71	ld (hl),c	bit 6,c	
114	r	72	ld (hl),d	bit 6,d	sbc hl,sp
115	s	73	ld (hl),e	bit 6,e	ld (NN),sp
116	t	74	ld (hl),h	bit 6,h	
117	u	75	ld (hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	ld (hl),a	bit 6,a	
12Ø	x	78	ld a,b	bit 7,b	in a,(c)
121	y	79	ld a,c	bit 7,c	out (c),a
122	z	7A	ld a,d	bit 7,d	adc hl,sp
123	{	7B	ld a,e	bit 7,e	ld sp,(NN)
124	|	7C	ld a,h	bit 7,h	
125	}	7D	ld a,l	bit 7,l	
126	~	7E	ld a,(hl)	bit 7,(hl)	
127	©	7F	ld a,a	bit 7,a	
128	□	8Ø	add a,b	res Ø,b	
129	▗	81	add a,c	res Ø,c	
13Ø	▖	82	add a,d	res Ø,d	
131	▄	83	add a,e	res Ø,e	
132	▝	84	add a,h	res Ø,h	
133	▐	85	add a,l	res Ø,l	
134	▞	86	add a,(hl)	res Ø,(hl)	
135	▟	87	add a,a	res Ø,a	
136	▘	88	adc a,b	res 1,b	
137	▚	89	adc a,c	res 1,c	
138	▌	8A	adc a,d	res 1,d	
139	▙	8B	adc a,e	res 1,e	
14Ø	▀	8C	adc a,h	res 1,h	
141	▜	8D	adc a,l	res 1,l	
142	▛	8E	adc a,(hl)	res 1,(hl)	
143	■	8F	adc a,a	res 1,a	
144	(a)	9Ø	sub b	res 2,b	
145	(b)	91	sub c	res 2,c	
146	(c)	92	sub d	res 2,d	
147	(d)	93	sub e	res 2,e	
148	(e)	94	sub h	res 2,h	
149	(f)	95	sub l	res 2,l	
15Ø	(g)	96	sub (hl)	res 2,(hl)	
151	(h)	97	sub a	res 2,a	
152	(i)	98	sbc a,b	res 3,b	
153	(j)	99	sbc a,c	res 3,c	
154	(k)	9A	sbc a,d	res 3,d	
155	(l)	9B	sbc a,e	res 3,e	
156	(m)	9C	sbc a,h	res 3,h	
157	(n)	9D	sbc a,l	res 3,l	
158	(o)	9E	sbc a,(hl)	res 3,(hl)	
159	(p)	9F	sbc a,a	res 3,a	
16Ø	(q)	AØ	and b	res 4,b	ldi
161	(r)	A1	and c	res 4,c	cpi
162	(s)	A2	and d	res 4,d	ini
163	(t)	A3	and e	res 4,e	outi
164	(u)	A4	and h	res 4,h	
165	RND	A5	and l	res 4,l	
166	INKEY$	A6	and (hl)	res 4,(hl)	
167	PI	A7	and a	res 4,a	
168	FN	A8	xor b	res 5,b	ldd

Gráficos definibles

CODIGO	CARACTER	HEX	ENSAMBLADOR DEL Z 8Ø	CB +...	ED +...
169	POINT	A9	xor c	res 5,c	cpd
17Ø	SCREEN$	AA	xor d	res 5,d	ind
171	ATTR	AB	xor e	res 5,e	outd
172	AT	AC	xor h	res 5,h	
173	TAB	AD	xor l	res 5,l	
174	VAL$	AE	xor (hl)	res 5,(hl)	
175	CODE	AF	xor a	res 5,a	
176	VAL	BØ	or b	res 6,b	ldir
177	LEN	B1	or c	res 6,c	cpir
178	SIN	B2	or d	res 6,d	inir
179	COS	B3	or e	res 6,e	otir
18Ø	TAN	B4	or h	res 6,h	
181	ASN	B5	or l	res 6,l	
182	ACS	B6	or (hl)	res 6,(hl)	
183	ATN	B7	or a	res 6,a	
184	LN	B8	cp b	res 7,b	lddr
185	EXP	B9	cp c	res 7,c	cpdr
186	INT	BA	cp d	res 7,d	indr
187	SQR	BB	cp e	res 7,e	otdr
188	SGN	BC	cp h	res 7,h	
189	ABS	BD	cp l	res 7,l	
190	PEEK	BE	cp (hl)	res 7,(hl)	
191	IN	BF	cp a	res 7,a	
192	USR	CØ	ret nz	set Ø,b	
193	STR$	C1	pop bc	set Ø,c	
194	CHR$	C2	jp nz,NN	set Ø,d	
195	NOT	C3	jp NN	set Ø,e	
196	BIN	C4	call nz,NN	set Ø,h	
197	OR	C5	push bc	set Ø,l	
198	AND	C6	add a,N	set Ø,(hl)	
199	<=	C7	rst Ø	set Ø,a	
200	>=	C8	ret z	set 1,b	
2Ø1	<>	C9	ret	set 1,c	
2Ø2	LINE	CA	jp z,NN	set 1,d	
2Ø3	THEN	CB		set 1,e	
2Ø4	TO	CC	call z,NN	set 1,h	
2Ø5	STEP	CD	call NN	set 1,l	
2Ø6	DEF FN	CE	adc a,N	set 1,(hl)	
207	CAT	CF	rst 8	set 1,a	
2Ø8	FORMAT	DØ	ret nc	set 2,b	
2Ø9	MOVE	D1	pop de	set 2,c	
21Ø	ERASE	D2	jp nc,NN	set 2,d	
211	OPEN #	D3	out (N),a	set 2,e	
212	CLOSE #	D4	call nc,NN	set 2,h	
213	MERGE	D5	push de	set 2,l	
214	VERIFY	D6	sub N	set 2,(hl)	
215	BEEP	D7	rst 16	set 2,a	
216	CIRCLE	D8	ret c	set 3,b	
217	INK	D9	exx	set 3,c	
218	PAPER	DA	jp c,NN	set 3,d	
219	FLASH	DB	in a,(N)	set 3,e	
22Ø	BRIGHT	DC	call c,NN	set 3,h	
221	INVERSE	DD	prefijos instruc ciones usando ix	set 3,l	
222	OVER	DE	sbc a,N	set 3,(hl)	
223	OUT	DF	rst 24	set 3,a	
224	LPRINT	EØ	ret po	set 4,b	
225	LLIST	E1	pop hl	set 4,c	
226	STOP	E2	jp po,NN	set 4,d	
227	READ	E3	ex (sp),hl	set 4,e	
228	DATA	E4	call po,NN	set 4,h	

CODIGO	CARACTER	HEX	ENSAMBLADOR DEL Z 8Ø	CB +...	ED +...
229	RESTORE	E5	push hl	set 4,l	
23Ø	NEW	E6	and N	set 4,(hl)	
231	BORDER	E7	rst 32	set 4,a	
232	CONTINUE	E8	ret pe	set 5,b	
233	DIM	E9	jp (hl)	set 5,c	
234	REM	EA	jp pe,NN	set 5,d	
235	FOR	EB	ex de,hl	set 5,e	
236	GO TO	EC	call pe,NN	set 5,h	
237	GO SUB	ED		set 5,l	
238	INPUT	EE	xor N	set 5,(hl)	
239	LOAD	EF	rst 4Ø	set 5,a	
24Ø	LIST	FØ	ret p	set 6,b	
241	LET	F1	pop af	set 6,c	
242	PAUSE	F2	jp p,NN	set 6,d	
243	NEXT	F3	di	set 6,e	
244	POKE	F4	call p,NN	set 6,h	
245	PRINT	F5	push af	set 6,l	
246	PLOT	F6	or N	set 6,(hl)	
247	RUN	F7	rst 48	set 6,a	
248	SAVE	F8	ret m	set 7,b	
249	RANDOMIZE	F9	ld sp,hl	set 7,c	
25Ø	IF	FA	jp m,NN	set 7,d	
251	CLS	FB	ei	set 7,e	
252	DRAW	FC	call m,NN	set 7,h	
253	CLEAR	FD	prefijos instruc- ciones usando iy	set 7,l	
254	RETURN	FE	cp N	set 7,(hl)	
255	COPY	FF	rst 56	set 7,a	

APENDICE B.- INFORMES

Estos aparecen al pié de la pantalla siempre que el computador se detiene al ejecutar un programa BASIC y explican porqué se ha detenido por una razón lógica o a causa de un error.

El informe empieza con un número o letra que es su código y cuyo significado veremos a continuación, y un breve mensaje explicando lo que ha sucedido y el número de línea o sentencia donde se ha detenido. (Una sentencia se indica como línea 0. Dentro de una misma línea, la sentencia 1 es la primera, la 2 la que está después de los dos puntos o después de THEN, etc..)

El comportamiento de CONTINUE depende sobremanera de los informes En condiciones normales, CONTINUE va a la línea y sentencia especificada en el último informe, pero hay excepciones con los informes 0, 9 y D (ver también Apendice C).

He aquí una tabla que detalla todos los informes. También le dice bajo qué circunstancias se ha producido éste y hace referencia al Apéndice C. Por ejemplo, el error A Invalid argument (Argumento no válido) puede presentarse con SQR, IN, ACS, y ASN y en el apéndice siguiente se le especificarán exactamente los argumentos que no son válidos.

CODIGO	SIGNIFICADO	Situaciones
0	O.K. Final feliz o salto a una línea mayor que las existentes. Este informe no cambia la linea y sentencia a la que se ha saltado, con el uso de CONTINUE.	Cualquiera
1	Sentencia NEXT sin el FOR correspondiente No existe la variable de control, pero existe una variable con el mismo nombre.	NEXT
2	Variable not found (Variable no encontrada). Para una variable simple esto se produce si dicha variable se emplea antes de que se haya asignado un valor a la misma mediante LET, READ, INPUT, o cargada desde casette o establecida en una sentencia FOR. Para una variable con subíndice se producirá si dicha variable es usada antes de que haya sido dimensionada con sentencia DIM o cargada desde el casette.	Cualquiera
3	Subscript wrong (subíndice erróneo). Un subíndice queda fuera de las dimensiones de la matriz o hay un número equivocado de subíndices. Si el subíndice es negativo o mayor de 65535, se producirá el error B.	Variables con subíndice. Subcadenas.
4	Out of memory (Fuera de memoria). No hay suficiente espacio en el computador para albergar lo que quiere entrarle. Si el computador parece realmente saturado en este estado, debe borrar la línea en curso usando DELETE y luego borrar una o dos líneas de programa(volviéndolas a colocar más tarde), para darle margen de maniobra ,con CLEAR por ejemplo.	LET,INPUT,FOR, DIM,GO SUB, LOAD,MERGE. A veces mientras evalúa expresiones.

CODIGO	SIGNIFICADO	SITUACIONES
5	Out of screen (Fuera de pantalla). Una sentencia INPUT ha tratado de generar más de 23 líneas en la parte inferior de la pantalla. También se produce con PRINT AT 22,...	INPUT, PRINT AT.
6	Number too big (Número demasiado grande). Los cálculos han generado un número superior a 10^{38} (aprox.).	En cualquier operación matemática.
7	RETURN without GO SUB (Return sin GO SUB). Hay mayor número de RETURN que de GO SUBs.	RETURN
8	End of file (fin del archivo).	Operaciones de Microdrive, etc
9	STOP Statement (Sentencia STOP). Después de esto, CONTINUE no repetirá el STOP sino que continuará con la siguiente sentencia.	STOP
A	Invalid Argument (Argumento no válido) El argumento de una función no sirve, por algún motivo.	SQR,LN,ASN,ACS, USR(con argumento de cadena).
B	Integer out of range (Entero fuera de margen) Cuando se requiere un entero, el argumento de la coma flotante se redondea al entero más próximo. Si éste se encuentra fuera de lo establecido, se produce el error B.	RUN,RANDOMIZE, POKE,DIM,GO TO, GO SUB,LIST, LLIST,PAUSE,PLOT CHR$,PEEK,USR (con argumento numérico).
	Para acceso a matrices ver también error 3	Acceso a matric.
C	Nonsense en BASIC (Sin sentido en BASIC). El texto del argumento no tiene expresión válida.	VAL, VAL$
D	BREAK-CONT repeats (BREAK, CONTINUE repite). Se ha pulsado BREAK durante la operación de un periférico. El comportamiento de CONTINUE tras este informe es normal, en el sentido de que repite la sentencia. Compare con el informe L.	LOAD,SAVE,VERIFY,MERGE,LPRINT, LLIST,COPY.También cuando el computador pregunta scroll? y se responde N SPACE o STOP.
E	Out of DATA (No hay más datos). Ha intentado probar READ después del final de una lista de DATA.	READ
F	Invalid file name (Nombre de archivo no válido). SAVE sin nombre o con uno superior a los 10 caracteres.	SAVE
G	No room for line (Sin espacio para la línea) No hay suficiente espacio en la memoria para albergar una nueva línea de programa.	Entrando una línea en el programa.
H	STOP in INPUT (Stop en entrada de datos) Algunos datos de INPUT empezaban con STOP o se ha pulsado esta tecla mientras se entraban los datos (INPUT LINE). Al contrario que con el informe 9, después del informe H, CONTINUE se comporta normalmente, repetiendo la sentencia INPUT.	INPUT

CODIGO	SIGNIFICADO	SITUACIONES
I	FOR without NEXT (FOR sin sentencia NEXT). Había un bucle FOR para no ser ejecutado (p.e. FOR n=1 TO 0)y no se ha encontrado, por tanto, el correspondiente NEXT.	FOR
J	Invalid I/O device (Entradas y salidas no válidas).	Operaciones con Microdrive, etc.
K	Invalid Colour (color no válido). El número especificado no tiene un valor adecuado.	INK,PAPER,BORDER, FLASH,BRIGHT,INVERSE,OVER,también después del correspondiente control de caracteres.
L	BREAK into program (BREAK en el programa) Se ha pulsado BREAK entre dos sentencias. El número de línea y sentencia del informe se réfiere a la sentencia anterior a pulsar BREAK, pero CONTINUE se dirigirá a la sentencia posterior(permitiendo efectuar cualquier salto) de manera que no se repite ninguna sentencia.	Cualquiera.
M	RAMTOP no good (RAMTOP incorrecto). El número especificado para RAMTOP es demasiado grande o demasiado pequeño.	CLEAR; quizás en RUN.
N	Statement lost (Sentencia extraviada). Salto a una sentencia que ya no existe.	RETURN,NEXT,CONTINUE.
O	Invalid Stream (canal no válido).	Operaciones con Microdrive, etc
P	FN without DEF (FN sin DEF). Función definida por el usuario.	FN
Q	Parameter error (error en parámetro) Número de argumentos equivocado o uno de ellos es de tipo incorrecto (de cadena en lugar de númérico o viceversa).	FN
R	Tape loading error (Error de carga de casette). Un archivo en cinta ha sido encontrado pero, por algún motivo, no ha sido cargado o bien no ha sido bien verificado (VERIFY).	VERIFY, LOAD o MERGE

APENDICE C.- DESCRIPCION RESUMIDA DEL SPECTRUM

La primera sección de este apéndice es una mera repetición del capítulo de Introducción en lo concerniente al teclado y a la pantalla.

EL TECLADO.-

Los caracteres del ZX Spectrum comprenden, además de los símbolos simples (letras, digitos, etc), los 'tokens' compuestos (comandos e instrucciones) con una simple pulsación de la tecla. Para conseguir todas estas funciones, determinadas teclas tienen cinco o más significados distintos, logrados mediante combinaciones de teclas (CAPS SHIFT ó SYMBOL SHIFT) y por los propios modos operativos del Spectrum.

El modo operativo viene indicado por el cursor, que es una letra parpadeante que indica la posición donde se insertará lo que se escriba en el teclado.

El modo K significa que la máquina está esperando una instrucción o un número de línea. Esta posición se produce al principio de línea o después de THEN o de ':'.

El modo L suele presentarse en los demás casos.

En los modos K y L, SYMBOL SHIFT y una tecla, proporcionarán el carácter escrito en rojo en dicha tecla. CAPS SHIFT y una letra proporcionarán la función de control escrita en blanco en esta tecla. CAPS SHIFT con las demás teclas no afecta a las palabras clave (keywords) en el modo K y, en modo L, convierte las minúsculas en mayúsculas.

El modo C es una variante del modo L y todas las letras aparecen en mayúscula. CAPS LOCK produce un cambio del modo L al C y viceversa.

El modo E se utiliza para obtener caracteres adicionales. Tiene lugar después de que se pulsen simultáneamente las teclas de SYMBOL SHIFT y CAPS SHIFT y dùra solamente una letra. Las dos teclas mencionadas juntas dan el comando superior en verde y, manteniendo una de ellas pulsada, dan el comando inferior en rojo.

El modo G se obtiene pulsando simultáneamente CAPS SHIFT y 9 (GRAPHICS) y permanece hasta que se repite la operación o se pulse simplemente 9. Una tecla de dígito proporcionará el gráfico indicado,(menos GRAPHICS ó DELETE) y cada tecla de letras (menos V,W,X,Y y Z), dará un gráfico definido por el usuario.

Manteniendo cualquier tecla apretada unos dos o tres segundos, se produce la autorepetición de la misma.

La introducción por el teclado se visualiza en la mitad inferior de la pantalla a medida que se teclea, insertándose cada carácter inmediatamente antes del cursor. Este puede desplazarse (el cursor) a la izquierda o a la derecha con los controles de dirección (CAPS SHIFT 5 y 8). El carácter precedente al cursor puede borrarse con DELETE (CAPS SHIFT y 0). La línea completa puede borrarse mediante EDIT (CAPS SHIFT y 1) y, a continuación,ENTER.

Cuando se pulsa ENTER, se ejecuta la línea, se introduce en el programa o se utiliza como entrada de datos cuando convenga, a menos que se produzca un error de sintaxis en cuyo caso aparece un ? parpadeante junto al error.

A medida que se entran las líneas del programa, se observa el listado que aparece en la mitad superior de la pantalla. La última línea introducida se denomina línea en curso y se indica con el símbolo ▶, éste puede desplazarse con el empleo de los controles de dirección

(CAPS SHIFT 6 y 7). Si se pulsa EDIT, lá línea en curso se reproduce en la parte inferior de la pantalla y puede corregirse, sustituyendo a la original, una vez corregida simplemente pulsando ENTER.

Cuando se ejecuta un comando o un programa, el resultado se visualiza en la mitad superior de la pantalla y se mantiene ahí hasta que se introduce una línea de programa o se pulsa ENTER con una línea vacía o se pulse SHIFT 6 6 7. En la parte inferior aparece un informe con un código al que se hace referencia en el Apéndice B. El informe apareoe en pantalla hasta que se pulse una tecla cualquiera (entrando en modo K).

En determinadas circunstancias, CAPS SHIFT junto con SPACE actúa como BREAK, deteniendo el computador con informe D o L. Ello se logra al final de una sentencia mientras se ejecuta un programa,o mientras el computador está empleando la impresora o el casette

LA PANTALLA.-

Tiene 24 líneas, con 32 caracteres cada una y está dividida en dos partes. La superior comprende, como máximo, 22 líneas y visualiza el listado o la ejecución del programa. Cuando la impresión de la parte superior, llega a la parte inferior, se desplaza hacia arriba (scroll) una línea ; si ello implicara la pérdida de una línea que no se hubiera examinado todavía, entonces el computador se detiene con el mensaje SCROLL?. Si pulsa las teclas N, SPACE o STOP hará que se interrumpa el programa con el informe D BREAK-CONT repeats, cualquier otra tecla que no sea las mencionadas, permitirá que continue el scrolling (o desplazamiento hacia arriba). La parte inferior de la pantalla se emplea para la introducción de comandos, líneas de programa o entrada de datos y, también, para visualizar informes que dá el computador. La parte inferior ocupa, en principio, las dos últimas líneas pero se amplía para admitir lo que pueda entrar desde el teclado. Cuando llegue al espacio destinado al listado, en la parte superior, ésta se desplazará hacia arriba para dejar sitio.

Cada posición de carácter tiene atributos que especifican los colores del PAPER (fondo) y del INK (texto), los dos niveles de brillo, y si parpadea o no. Los colores disponibles son negro, azul, rojo, magenta, verde, azul claro, amarillo y blanco.
El contorno de la pantalla puede disponerse de cualquier color mediante el comando BORDER.

Una posición de carácter se divide en 8x8 pixels y se obtienen gráficos de alta resolución asignándo a cada pixel individualmente el color para el fondo o el texto de cada posición de carácter.

Los atributos de cada posición se ajustan siempre que se imprime un carácter o se posiciona un pixel. La forma exacta de ajuste se determina mediante los parámetros de impresión que pueden ser de dos tipos : permanentes o temporales, con 6 parámetros diferentes PAPER, INK, FLASH, BRIGHT, INVERSE y OVER. Los parámetros permanentes de la parte superior se establecen mediante PAPER, INK, etc.. y permanecen hasta nueva orden.(Inicialmente se establece negro sobre fondo blanco, con brillo normal, sin parpadeo, video normal y sin sobreimpresión). Los parámetros permanentes de la parte inferior de la pantalla emplean el color del contorno como color de fondo, contrastando con colores blanco o negro para la tinta (INK), brillo normal, sin parpadeo, video normal sin sobreimpresión.

Los parámetros temporales se establecen mediante las mismas partículas PAPER, INK, etc.. que están incluidas en las sentencias PRINT, LPRINT, INPUT, PLOT, DRAW y CIRCLE y también en los caracteres de control PAPER, INK, etc.. que, cuando van a imprimirse en la pantalla deben ir seguidos por un byte adicional para especificar el valor del parámetro.. Los parámetros temporales duran solamente hasta el final

de la sentencia PRINT (o la que sea) o, en sentencias INPUT, hasta que el dato de dicho INPUT ha sido entrado por el teclado, cuando ha sido reemplazado por los parámetros permanentes.

Los parámetros PAPER e INK están en la escala de 0 a 9. Los parámetros de 0 a 7 son los colores empleados cuando se imprime un caracter :

0 negro
1 azul
2 rojo
3 magenta
4 verde
5 cian
6 amarillo
7 blanco

El parámetro 8 ('transparente') especifica que el color de la pantalla permanecerá sin cambios cuando se imprima un carácter.
El parámetro 9 ('contraste') especifica que el color en cuestión (papel o tinta) será blanco o negro para que resalte contra el otro color.

Los parámetros FLASH y BRIGHT valen 0, 1 u 8 ; 1 significa que el brillo o el parpadeo están en vigor, 0 que están desactivados y 8 que no cambiarán el estado del carácter anterior.

Los parámetros OVER e INVERSE tienen valor 0 ó 1.

OVER 0 los caracteres nuevos eliminan a los anteriores.
OVER 1 las configuraciones de bits del viejo y nuevo carácter se combinan mediante operación OR exclusiva (sobreimpresión).
INVERSE 0 los caracteres nuevos se imprimen en color tinta sobre color fondo (video normal).
INVERSE 1 los caracteres nuevos se imprimen en negativo, color fondo sobre color tinta (video inverso).

Cuando el parámetro de control TAB se recibe en el televisor, se esperan dos bytes adicionales para especificar el punto n de paro de TAB (byte de menor peso en primer lugar). Este se reduce a módulo 32 a n_0 (por ejemplo) y entonces se dejan suficientes espacios para mover la posición de PRINT hasta la columna n_0.

Cuando se recibe una coma como control de caracteres, se imprimen suficientes espacios (1 como mínimo) para mover la posición de PRINT hasta la columna 0 hasta la 16.

Cuando se recibe un ENTER, la posición de PRINT se desplaza hasta la siguiente línea.

LA IMPRESORA.-

La salida hacia la impresora ZX se efectúa mediante un buffer de una línea (32 caracteres) y dicha línea es enviada a la impresora,

a) cuando se completa una línea,
b) cuando se recibe una orden ENTER,
c) al final de un programa, si algo permanece sin imprimir,
d) cuando un control TAB o una coma mueve la posición de impresión hasta una nueva línea.

Los controles TAB y coma generan espacios en la impresora de la misma manera que en la pantalla.

El control AT cambia la posición de impresión empleando el número de columna e ignorando el número de línea.

La impresora responde a los comandos INVERSE y OVER en forma similar a la pantalla, pero es insensible a PAPER, INK, FLASH o BRIGHT.

La impresora se detendrá con error B si se pulsa BREAK.

Si la impresora no está conectada todos los datos, obviamente, se desperdiciarán.

EL SISTEMA BASIC

Los números se almacenan con una precisión de 9 ó 10 dígitos. El mayor número que puede manejar es aproximadamente 10^{38} y el menor (positivo) alrededor de $4*10^{-39}$.

Un número se almacena en el ZX Spectrum con coma flotante en binario con un exponente de 1 byte e ($1 <= e <= 255$) y una mantisa de cuatro bytes m ($1/2 <= m < 1$). Esto representa el número $m*2^{e-128}$.

Dado que $1/2 <= m < 1$, el bit de mayor peso de la mantisa m es siempre 1. Por tanto realmente podemos reemplazarlo con un bit para señalar el signo (0 para números positivos, 1 para negativos).

Los enteros pequeños tienen una representación especial en la que el primer byte es 0, el segundo es el byte de signo (0 ó FFh) y el tercero y el cuarto están en forma complemento de dos, con el byte de menor peso en primer lugar.

Las variables numéricas tienen nombres de longitud arbitraria, empezando con una letra y continuando con letras o números. Los controles de colores y espacios se ignoran y todas las letras se interpretan como minúsculas.

Las variables de control de los bucles FOR-NEXT tienen el nombre formado por una letra única.

Las matrices numéricas tienen nombres de una sola letra, que puede coincidir con la de una variable simple. Pueden tener arbitrariamente varias dimensiones de cualquier medida. Los subíndices empiezan a partir de 1.

Las cadenas pueden tener cualquier longitud. El nombre de una cadena lo forma una sola letra seguida de $.

Las matrices de cadenas pueden tener arbitrariamente muchas dimensiones de cualquier medida. El nombre es una letra sola seguida de $ y no debería coincidir con el nombre de una cadena. Todas las cadenas de una determinada matriz tienen exactamente la misma longitud fijada, la cual se especifica con una dimensión adicional en la sentencia DIM. Los subíndices empiezan a partir de 1.

'Slicing'(fragmentado) : Pueden disgregarse subcadenas de cadenas completas empleando 'slicers'(fragmentadores). Un 'slicer' puede ser:

a) vacío
b) expresión numérica
c) expresión numérica opcional TO (a) otra expresión numérica opcional.

y se emplea,para expresar una subcadena, en la forma :

1) expresión de cadena (slicer)
2) variable de matriz de cadena (subíndice,..., subíndice, slicer).

que equivale a lo mismo que

variable de matriz de cadena (subíndice....., subíndice)(slicer)

En el caso 1), supongamos que la expresión de cadena tiene el valor s$.

Si el slicer es vacío, el resultado es s$ considerada como una subcadena de sí misma.

Si el slicer es una expresión numérica de valor m, entonces el resultado es el m-ésimo carácter de s$ (una subcadena de longitud 1)

Si el slicer tiene la forma del apartado c), entonces supongamos que la primera expresión numérica tiene un valor m (el valor por defecto es 1) y la segunda, un valor n (el valor por defecto es la longitud de s$).

Si $1 <= m <= n <=$ a la longitud de s$, entonces el resultado es la subcadena de s$ que empieza en el m-ésimo carácter y termina en el n-ésimo.

Si $0 <= n < m$ entonces el resultado es la cadena vacía.

En cualquier otro caso se produce el error 3.

La fragmentación (slicing) se efectúa antes de evaluar las funciones y operaciones, a menos que los paréntesis lo indiquen de otra forma.

Las subcadenas pueden ser asignadas (ver LET).

Si unas comillas de cadena deben de escribirse en el interior de un literal de cadena, éstas (las comillas) deben doblarse.

FUNCIONES .-

El argumento de una función no necesita paréntesis si se trata de una constante o de una variable (con subíndices o fragmentada, posiblemente).

Función	Tipo de argumento (x)	Resultado
ABS	número	Magnitud absoluta
ACS	número	Arcoseno en radianes. Error A si x no está en el rango de - 1 a +1.
AND	operación binaria el término de la derecha siempre un número. Término de la izquierda numérico :	A AND B = { A si B<>0 ; 0 si B=0 }
	Término de la izquierda una cadena :	A$ AND B = { A$ si B<>0 ; "" si B=0 }
ASN	número	Arcoseno en radianes: Error A si x no está en el rango de -1 a +1.
ATN	número	Arcotangente en radianes.
ATTR	dos argumentos, x e y, ambos números encerrados en paréntesis.	Un número cuya forma binaria indica en código los atributos de la línea x y columna y en el televisor. El bit 7 (de mayor peso) vale 1 para parpadeo y 0 para fijo. Bit 6 es 1 para brillo y 0 para normal. Los bits del 5 al 3 indican el color del fondo (papel). Los bits 2, 1 y 0 indican el color de la tinta (INK). Aparece el error B si no se cumple $0 <= x <= 23$ y $0 <= y <= 31$

BIN		No se trata en realidad de una función, sino una notación alternativa para números : BIN seguido de una secuencia de Øs y 1s es el número correspondiente expresado en forma binaria.
CHR$	número	El carácter cuyo código es x redondeado al entero más próximo.
CODE	cadena	El código del primer carácter de x (o Ø si x es una cadena vacía).
COS	número (en radianes)	Coseno de x
EXP	número	e^x (e elevado a x).
FN		FN seguido por una letra designa a una función definida por el usuario (ver DEF). Los argumentos deben estar encerrados entre paréntesis; estos paréntesis deben indicarse aunque no existan argumentos.
IN	número	El resultado es una entrada de datos a nivel del procesador desde la puerta x (Ø <= x <= FFFFh)(carga el par de registros bc con x y realiza la instrucción en assembler de a(c)).
INKEY$	ninguno	'Lee' el teclado . El resultado es el carácter que representa (en modo L ó C) la tecla pulsada en este momento y, caso de haber varias, la cadena vacía.
INT	número	Parte entera (siempre redondea por abajo)
LEN	cadena	Longitud.
LN	número	Logaritmo natural (en base e). Error A en el caso de que x =Ø.
NOT	número	Ø si x<>Ø, 1 si x=Ø. NOT tiene prioridad 4.
OR	operación binaria. Ambos términos són números.	a OR b= { 1 si b<>Ø ; a si b=Ø } OR tiene prioridad 2
PEEK	número	El valor del byte de memoria cuya dirección es x (redondeada al entero más cercano). Se produce error B si x no está en el rango de Ø a 65535.
PI	ninguno	π (3.14159265...)
POINT	dos argumentos, x e y, ambos números; entre paréntesis.	1 si el pixel (x,y) es color tinta (INK) y Ø si es el color del fondo (PAPER)
RND	ninguno	El próximo número pseudoaleatorio de una secuencia generada tomando potencias de 75, módulo 65537, restándole 1 y dividiendo por 65536. Ø <= y < 1.
SCREEN$	Dos argumentos numéricos, x e y encerrados en paréntesis	El carácter que aparece en forma normal o invertida en la pantalla en la línea x y columna y. Dá la cadena vacía si no se reconoce el carácter. Se produce error B no se cumple Ø <= x <= 23 y Ø <= y <= 31

Función	Tipo de argumento (x)	Resultado
SGN	número	Signo : el signo (-1 para negativo, Ø para cero o +1 para positivo)de x.
SIN	número (en radianes)	Seno de x
SQR	número	Raiz cuadrada. Error A si x<Ø.
STR$	número	La cadena de caracteres que se vería en pantalla si se imprime x.
TAN	número(en radianes)	Tangente
USR	número	Llama a la subrutina en código máquina que empieza en la dirección x. De retorno, el resultado es el contenido del par de registros bc.
USR	cadena	La dirección del conjunto de bits que forman el gráfico predefinido que corresponde a x. Se produce el error A si x no es una letra entre a y u o un gráfico predefinido por el usuario.
VAL	número	Evalúa x como una expresión numérica. Se produce el error C si x tiene un error de sintaxis o representa un valos de una cadena. Hay otros errores posibles según sea la expresión.
VAL$	cadena	Evalúa x como una expresión de cadena. Se produce el error C si x contiene un error de sintaxis o dá un valor numérico. Otros errores son posibles, al igual que en VAL.
-	número	Negación.

Lo que sigue, se trata de operaciones binarias :

+	Adición o suma (con números), o concatenación (con cadenas).	
-	Sustracción o resta.	
*	Multiplicación.	
/	División.	
↑	Elevar a una potencia. Error B si el término de la izquierda es negativo.	
=	Igual.	Ambos términos deben ser del mismo tipo. El resultado es el número 1 si la comparación es correcta y Ø en caso contrario.
>	Mayor que	
<	Menor que	
<=	Menor o igual que	
>=	Mayor o igual que	
<>	No igual a (distinto de)	

Las funciones y las operaciones tienen las siguientes prioridades :

Operación	Prioridad
Asignación de subíndices y 'slicing'	12
Todas las funciones excepto NOT y aquellas que representen algún tipo de negación.	11
	10
Operaciones en las que el signo menos sirva para negar algo.	9
*, /	8
+,- (el signo menos empleado para restar un	

Operación	Prioridad
número de otro.	6
=, >, <, <=, >=, <>	5
NOT	4
AND	3
OR	2

Sentencias.-

En la siguiente lista :

a	representa una letra simple.
v	representa una variable.
x,y,z	representan expresiones numéricas.
m,n	representan expresiones numéricas redondeadas al entero más próximo.
e	representa una expresión.
f	representa una expresión valorada de cadena.
s	representa una secuencia de sentencias separadas separadas por los dos puntos (:).
c	representa una secuencia de partículas de color, cada una terminada en comas o punto y coma (, ó ;).Una partícula de color se expresa con las sentencias PAPER, INK, FLASH, BRIGHT, INVERSE u OVER.

Observe que las expresiones arbitrarias se permiten en cualquier parte excepto en el número de línea al comienzo de una sentencia.

Todas las sentencias, a excepción de INPUT, DEF y DATA pueden emplearse indistintamente como comandos o en medio de programas (aunque varíen ligeramente según sea el caso). Un comando o línea de programa puede tener varias sentencias, separadas por los dos puntos (:). No hay distinción respecto a la colocación de una determinada sentencia en una línea,(no obstante vea IF y REM).

BEEP x,y — Genera una nota a través del altavoz durante x segundos con un tono y medios tonos por encima del DO (o por debajo si y es negativo).

BORDER m — Establece el color del contorno de la pantalla así como el color del fondo de la parte inferior de la misma. Se produce error K si m no está entre Ø y 7.

BRIGHT n — Establece el brillo de los caracteres impresos a continuación. n= Ø para normal, 1 para brillo, 8 para transparente. Error K si n no es Ø, 1 u 8.

CAT — No tiene efecto sin Microdrive conectado.

CIRCLE x,y,z — Traza un arco de cicunferencia de centro (x,y),radio z.

CLEAR — Borra todas las variables, dejando libre el espacio que ocupaban. Eféctúa RESTORE y CLS, repone la posición de PLOT en la esquina inferior izquierda y borra la pila de GO SUB.

CLEAR n — Igual que CLEAR pero, si es posible, cambia la variable del sistema RAMTOP a n y coloca ahí la nueva pila de GO SUB.

CLOSE # — No tiene efecto sin Microdrive conectado.

CLS — (clear screen). Borra pantalla. Borra fichero de pantalla.

CONTINUE — Continúa el programa, empezando a partir de donde se detuvo con informe que no fuera Ø. Si el informe fué 9 ó L, entonces continúa con la siguiente sentencia (teniendo en cuenta posibles saltos); en caso contrario repite la sentencia donde se produjo el error .
Si el informe anterior se produjo en una línea, entonces CONTINUE intentará continuar esta línea y entrará en un bucle si el error fué Ø:1, dará informe Ø si fué Ø:2, o dará error N si el informe fué Ø:3 o superior. CONTINUE aparece en el teclado como CONT.

COPY — Manda una copia de las 22 líneas de la pantalla a la impresora si está conectada; de lo contrario no tiene efecto. Observe que COPY no tiene efecto (no reproduce) los informes o listados automáticos que puedan aparecer al pié de la pantalla. Si pulsa BREAK aparece el informe D.

DATA $e_1,e_2,e_3,..$ — Parte de la lista de datos (DATA).Debe estar en el programa.

DEF FN $a(a_1,..,a_k)=e$ — Forma de la función pre-definida; debe estar en el programa. Tanto a como $a_1 \ldots a_k$ deben ser una letra única o bien una letra seguida de $ para argumento de cadena o resultado. Toma la forma DEF FN a() si no hay argumentos.

DELETE f — No tiene efecto sin Microdrive conectado.

DIM $a(n_1,..,n_k)$ — Elimina cualquier matriz con el nombre a y establece una nueva de nombre a numérica con k dimensiones $n_1,\ldots,n_k$.Todos los valores valen Ø al empezar.

DIM $a\$(n_1,..,n_k)$ — Borra cualquier matriz o cadena de nombre a$ y establece una nueva matriz de caracteres con k dimensiones $n_1,\ldots,n_k$. Inicializa todos los valores a " ". Puede considerarse como una matriz de cadenas de longitud determinada n_k, con k-1 dimensiones $n_1,\ldots,n_{k-1}$.
Se produce el error 4 si no hay espacio para alojar la matriz. Una matriz es indefinida hasta que se dimensiona con la sentencia DIM.

DRAW x,y — DRAW x,y,Ø.

DRAW x,y,z — Traza una línea desde la actual posición de PLOT moviéndose x horizontalmente e y verticalmente con relación respecto a ésta, mientras gira con un ángulo z.
Se produce el error B si se sale fuera de la pantalla.

ERASE — No tiene efecto sin el Microdrive conectado.

FLASH — Define si el cárácter debe permanecer fijo o 'parpadeante'. n=Ø para fijo, n=1 para parpadeo y n=8 para permanecer tal como estaba.

FOR a=x TO y — FOR a=x TO y STEP 1.

FOR a=x TO y STEP z — Borra cualquier variable simple a y establece una variable de control de valor x, límite y y paso z y dirección de bucle repetitivo respecto a la sentencia que sigue a FOR. Comprueba si el valor inicial es mayor (si el paso > =Ø) o menor (paso <Ø) que el límite y si es correcto salta a la sentencia NEXT a, dando error 1 si no existe. Ver NEXT. Se produce error 4, si no hay espacio para la variable de control.

FORMAT f — No tiene efecto sin Microdrive conectado.

GO SUB n — Empuja el número de línea de la sentencia GO SUB en una pila; entonces actúa como GO TO n. Se produce error 4 si no hay suficientes RETURNs.

GO TO n — Salta a la línea n (ó, si ésta no existe, a la primera línea después de ésta).

IF x THEN s — Si x es 'verdad' (distinta de Ø) se ejecuta s. Nótese que s comprende todas las sentencias hasta el final de la línea. No se permite la forma IF x THEN nºde línea.

INK n — Establece el color de la tinta (texto) de los caracteres que van a imprimirse. n está entre Ø y 7 para el color, n=8 para transparente y n=9 para contraste. Ver el apartado LA PANTALLA en este mismo apéndice. Se produce error K si n no está comprendida entre Ø y 9

INPUT... — Los puntos '...' son una secuencia de partículas de INPUT separadas, al igual que en PRINT por comas, punto y coma o apóstrofes. Una partícula INPUT puede ser

a) Cualquier partícula PRINT que no empiece con una letra

b) El nombre de una variable, o

c) LINE seguido por el nombre de una variable del tipo cadena.

Las partículas PRINT y los separadores del apartado a) son tratados exactamente igual que en PRINT, excepto en que todo se imprime al pié de la pantalla

En el caso de b) el computador se detiene y espera la entrada de una expresión desde el teclado, el valor de ésta se asigna a la variable. La entrada se dispone de la forma habitual y los posibles errores de sintaxis provocan el ? parpadeante. Para expresiones del tipo cadena; el buffer de entrada se dispone para contener dobles comillas (que pueden borrarse si se desea). Si el primer carácter en el INPUT es STOP, el programa se detiene con el error H. El apartado c) es igual que el b) excepto en que el input o entrada es tratado como un literal de cadena sin comillas y el mecanismo de STOP no funciona en este caso; para detenerlo debe entrar ♦.

INVERSE n — Controla la inversión de los caracteres que van a imprimirse. Si n=Ø, los caracteres se imprimen en vídeo normal, color tinta sobre color fondo. Si n=1, los caracteres se imprimen en vídeo inverso (en negativo), es decir, color fondo sobre color tinta. Ver apartado de 'La Pantalla' en este mismo apéndice. Se produce el error K si n es distinto de Ø ó de 1.

LET v=e — Asigna el valor de e a la variable v. La expresión LET no puede omitirse. Una variable simple es indefinida hasta que se le asigna un valor mediante sentencia LET, READ o INPUT. Si v es una varible de cadena con subíndice, o una variable de una cadena fragmentada (subcadena), entonces la asignación es Procrusteana (longitud fija): el valor de cadena de e es truncado o llenado mediante espacios a su derecha para hacerla de la misma longitud que la variable v.

LIST — LIST Ø.

LIST n — Realiza el listado del programa en la parte superior de la pantalla, empezando en la primera línea después de n (si n no está) y dejando el cursor del programa en dicha línea.

LLIST — LLIST Ø.

LLIST n — Idéntico a LIST n, pero usando la impresora.

LOAD f — Carga programas y variables desde el casette.

LOAD f DATA() — Carga matrices numéricas.

LOAD f DATA $() — Carga caracteres de matriz de cadena.

LOAD f CODE m,n — Carga hasta n bytes empezando en la dirección m.

LOAD f CODE m — Carga bytes empezando en la dirección m.

LOAD f CODE — Carga de vuelta bytes a la dirección en que estaban guardados antes de hacer SAVE.

LOAD f SCREEN$ — LOAD f CODE 16384,6912. Busca el fichero de la clase correspondiente en el casette y lo carga, borrando anteriores versiones, en la memoria. Ver capítulo XX

LPRINT — Igual que PRINT empleando la impresora.

MERGE f — Igual que LOAD f, pero sin borrar las antiguas líneas ni variables que estuvieran en el computador, a menos que coincidan con las que se están cargando, caso en que prevalece lo nuevo.

MOVE f_1,f_2 — No tiene efecto sin MICRODRIVE conectado.

NEW — Empieza de nuevo el sistema BASIC, borrando programa y variables, usando el máximo de memoria, e incluyendo el byte cuya dirección está en la variable del sistema RAMBOT, preservando las variables del sistema UDG, P RAMT, RASP y PIP.

NEXT a — a) Encuentra la variable de control a
b) Añade el paso a su valor (STEP).
c) Si el(paso) step >=Ø y el valor > el límite; o si el paso <Ø y el valor < el límite, entonces salta a la sentencia del bucle.
Se produce error 2 si no hay variable a. El error 1 se produce si hay variable pero no es a.

OPEN # — No tiene efecto sin Microdrive conectado.

OUT m,n — Produce la salida del byte n en la puerta m a nivel de procesador (carga el par de registros bc con m, el registro a con n y efectúa la instrucción en lenguaje assembler : out (c),a.). Se produce error B si no se cumple Ø <=m <=65535 y -255 <=n <=255.

OVER n — Controla la sobreimpresión de los caracteres que van a imprimirse. Si n=Ø, los caracteres eliminan a los anteriores en esa posición. Si n=1, entonces los nuevos caracteres se mezclan con los viejos para dar color de tinta donde cada uno (pero no los dos juntos) tenía color de tinta y color papel si ambos tenían el mismo color de papel. Ver 'La Pantalla' en este apéndice. Se produce error K si n es distinto de Ø ó de 1.

PAPER n — Igual que INK, pero controlando el color papel (fondo).

PAUSE n — Detiene la computación y muestra el fichero de pantalla durante n ciclos (50 por segundo en nuestro país) o

hasta que se pulsa una tecla. Se produce error B si no se cumple $0 <= n <= 65535$. Si n=0 entonces la pausa es indefinida hasta que se pulsa una tecla.

PLOT c;m,n — Imprime una mácula de tinta (sujeta a OVER e INVERSE) en el pixel (m,n); desplaza la posición de PLOT.
A menos que la partícula de color c especifique lo contrario, el color de tinta de la posición de carácter que contiene el pixel se sustituye por el color permanente actual (en este momento) y los otras partículas (color papel, parpadeo y brillo) permanecen sin cambiar. Se produce error B si no se cumple : $0 <= m <= 255$, $0 <= n <= 255$.

POKE m,n — Asigna el valor n al byte almacenado cuya dirección es m. Error B si no se cumple $0 <= m <= 65535$, $-255 <= n <= 255$.

PRINT... — Los puntos '...' indican una secuencia de partículas de PRINT, separadas por comas (,), punto y coma(;) o apóstrofes (') que se instalan en el fichero de pantalla para aparecer en la pantalla.
Un punto y coma (;) entre dos partículas no tiene efecto alguno ; se emplea simplemente para separar los términos. Una coma (,) genera el control de carácter correspondiente y un apóstrofe (') provoca el carácter ENTER.
Al final de una sentencia PRINT, si no hay ninguno de los tres símbolos citados (;,'), se genera un carácter ENTER. Un término o partícula PRINT puede ser :

a) vacío, es decir, nada.
b) una expresión numérica
En primer lugar se imprime un signo - si el valor es negativo. Asignemos x= módulo de valor.
Si $x <= 10^{-5}$ o $x >= 10^{13}$, entonces se imprime usando notación científica. La parte mantisa tiene un máximo de ocho dígitos (sin ceros a la derecha) y la coma decimal (en el caso de más de un dígito) está a continuación del primero. La parte exponente es E, seguida de + ó - y un número de uno o dos dígitos.
En cualquier otro caso x se imprime en notación decimal normal con un máximo de 8 dígitos significativos y sin ceros después de la coma decimal. Si la coma decimal se encuentra justo al comienzo de un número, siempre va seguida por un 0. Por ejemplo : .03 , 0.3
El 0 se imprime como un único dígito 0.
c) una expresión de cadena
Los comandos en la cadena están expandidos, quizás con espacios antes o después de los mismos.
Los caracteres de control tienen su efecto normal.
Los caracteres inidentificables se imprimen como ?.
d) AT m,n
Genera un control de carácter AT seguido de un byte para m (número de línea) y otro para n (número de columna).
e) TAB n
Genera un control de carácter TAB seguido por dos bytes para n (el byte de menor peso primero) que es el stop de TAB.
f) Una partícula de color que adopta la forma de sentencia PAPER, INK, FLASH, BRIGHT, INVERSE u OVER.

RANDOMIZE — RANDOMIZE 0

RANDOMIZE n — Establece la variable del sistema (llamada SEED) empleada para generar el siguiente valor de RND. Si n<>0, SEED toma el valor n; si n=0 entonces adopta el valor de otra variable del sistema (llamada FRAMES) que cuenta los cuadros (FRAME=cuadro) que se han generado en la pantalla (a razón de 50 cuadros/seg.) desde la conexión del computador y, por tanto, es un número suficientemente aleatorio.. RANDOMIZE aparece como RAND en el teclado.
Se produce error B si n no está entre 0 y 65535.

READ $v_1,v_2,..,v_k$ — Asigna a las variables sucesivas expresiones contenidas en la lista de DATA..
Se produce error C si una expresión es incorrecta.
Se produce error E si quedan variables todavía por 'leer' cuando se agote la lista de DATA.

REM... — Sin efecto.'...' puede ser cualquier secuencia de caracteres con excepción de ENTER. Estos incluyen los dos puntos (:), por lo que no se puede poner ninguna otra sentencia a continuación de REM en la misma línea.

RESTORE — RESTORE 0

RESTORE n — Regresa el puntero de DATA a la primera sentencia de DATA de una línea cuyo número sea n (o la siguiente si n no existe): la próxima sentencia READ empezará ahí a 'leer'.

RETURN — Toma referencia de la última sentencia de la pila de GO SUB y salta a la línea que la sigue. Se produce el error 7 si no hay sentencia de referencia en la pila :se trata de un error en el programa ; el número de GOSUBs no se corresponde con el de RETURNs.

RUN — RUN 0

RUN n — CLEAR y luego GO TO n.

SAVE f — Salva (guarda en casette) el programa y las variables.

SAVE f LINE — Salva el programa y las variables de tal forma que al cargarlo de nuevo ejecuta automáticamente el programa desde su comienzo.

SAVE f LINE m — Salva el programa y sus variables de forma que al cargarlo de nuevo se ejecuta automáticamente a partir de la línea m.

SAVE f DATA() — Salva la matriz numérica.

SAVE f DATA $() — Salva la matriz de caracteres $.

SAVE f CODE m,n — Salva n bytes a partir de la dirección m.

SAVE f SCREEN$ — SAVE f CODE 16384,6912. Salva información en casette, dándole el nombre f. Se produce error F si f es vacía o tiene más de 10 caracteres. Ver capítulo XX.

STOP — Detiene el programa con informe 9. CONTINUE seguirá en la próxima sentencia.

VERIFY — Lo mismo que LOAD excepto que los datos no se cargan en la RAM, sino que se comparan con los que ya hay. Se produce error R si los bytes no coinciden en la comparación.

APENDICE E.- BINARIO Y HEXADECIMAL

Este apéndice describe la forma de contar del computador, empleando el sistema binario.

La mayoría de los lenguajes de Europa emplean una serie más o menos regular de decenas para contar que, aunque en sus inicios fué bastante informal, actualmente se ha estabilizado en grupos regulares :

veinte, veintiuno, veintidós,....., veintinueve
treinta, treinta y uno, treinta y dos,...., treinta y nueve
cuarenta, cuarenta y uno,....., cuarenta y nueve

y así sucesivamente, gracias a los números árabes. No obstante, la única razón de usar decenas es que el ser humano tiene diez dedos.

En lugar de emplear el sistema decimal, de base diez, los computadores emplean una forma de binario llamada hexadecimal (hex para abreviar), basada en el número 16. Dado que sólo tenemos 1Ø números en nuestro sistema, necesitamos 6 dígitos adicionales para realizar el conteo. Así que usaremos A,B,C,D y F. Y que viene después de F ?.Al igual que nosotros, con diez dedos, escribimos 1Ø para diez, los computadores escriben 1Ø para dieciseis. Su sistema numeral es del siguiente modo :

Hex	Normal(base 10)
Ø	cero
1	uno
2	dos
3	tres
4	cuatro
5	cinco
6	seis
7	siete
8	ocho
9	nueve

y aquí termina el parecido, continuando...

A	diez
B	once
C	doce
D	trece
E	catorce
F	quince
1Ø	dieciseis
11	diecisiete
:	:
19	veinticinco
1A	veintiseis
1B	veintisiete
.	.
.	.
.	.
1F	treinta y uno
2Ø	treinta y dos
21	treinta y tres
.	.
.	.

.	.
.	.
9E	ciento cincuenta y ocho
9F	ciento cincuenta y nueve
AØ	ciento sesenta
A1	ciento sesenta y uno
.	.
.	.
B4	ciento ochenta
.	.
.	.
FE	doscientos cincuenta y cuatro
FF	doscientos cincuenta y cinco
1ØØ	doscientos cincuenta y seis

Si emplea notación hex y quiere simplificar las cosas, escriba 'h' al final de un número y diga 'hex'. Por ejemplo, para ciento cincuenta y ocho, escriba '9Eh' y pronuncie 'nueve E hex'.

Estará pensando que tiene que ver todo esto con los computadores. De hecho, éstos se comportan como si sólo tuvieran dos dígitos, representados por una baja tensión u 'off' (Ø) y por una alta tensión u 'on'(1). Esto se llama sistema binario, y los dos dígitos de este sistema llamados 'bits' : así pues un bit puede valer Ø ó 1.

En resumen, éstos son los primeros números de los sistemas que hemos mencionado.

español	decimal	hexadecimal	binario
cero	Ø	Ø	Ø ó ØØØØ
uno	1	1	1 ó ØØØ1
dos	2	2	1Ø ó ØØ1Ø
tres	3	3	11 ó ØØ11
cuatro	4	4	1ØØ ó Ø1ØØ
cinco	5	5	1Ø1 ó Ø1Ø1
seis	6	6	11Ø ó Ø11Ø
siete	7	7	111 ó Ø111
ocho	8	8	1ØØØ
nueve	9	9	1ØØ1
diez	1Ø	A	1Ø1Ø
once	11	B	1Ø11
doce	12	C	11ØØ
trece	13	D	11Ø1
catorce	14	E	111Ø
quince	15	F	1111
dieciseis	16	10	1ØØØØ

La característica importante es que dieciseis es igual a dos elevado a la cuarta potencia (16= 2↑4) y ello hace fácil la conversión entre hex y binario.

Para convertir hex a binario, cambie cada dígito en hex por cuatro bits, usando la tabla anterior. Para convertir binario en hex, divida el número binario en grupos de cuatro bits, empezando por la derecha, y luego cambie cada grupo por su correspondiente equivalente en hex.

Por esta razón, a pesar de que, estrictamente hablando, los computadores emplean binario puro, los humanos podemos guardar números en el interior del computador empleando hex.

Los bits en el interior del computador, se agrupan en series de 8

llamadas bytes. Un simple byte puede representar cualquier número entre Ø y 256 (11111111 en binario ó FF en hex), o también cualquier carácter del set que posee el Spectrum. Su valor puede escribirse mediante dos dígitos hex.

Dos bytes pueden agruparse para formar lo que técnicamente se llama una 'palabra'. Una palabra puede escribirse empleando dieciseis bits o cuatro dígitos hex y representa un número (en decimal) de Ø a 216-1 = 65535.

Un byte siempre lo forman ocho bits pero una palabra puede variar en longitud según el computador.

La notación BIN en el capítulo XIV proporciona la manera de escribir números en binario en el ZX Spectrum : BIN Ø representa cero, BIN 1 significa 1, BIN 1Ø significa 2, etc..

Sólo se pueden usar Øs y 1s en este sistema, así que los números deben ser enteros y no-negativos; por ejemplo, no puede escribir 'BIN -11' en lugar de menos tres sino que debe poner -BIN 11. El número no debe ser mayor del decimal 65535 ya que no puede tener más de dieciseis bits.

ATTR es, en realidad, binario. Si convierte el resultado de ATTR en binario, puede escribirlo con ocho bits.

El primero es 1 para parpadeo ó Ø para fijo.
El segundo es 1 para brillo,Ø para normal.
Los tres siguientes son el código para el color del papel, escrito en binario.
Los tres últimos son el código para el color tinta, en binario.

Los códigos de colores también emplean el binario : cada código escrito en binario puede hacerse con tres bits, el primero para el verde, el segundo para el rojo y el tercero para el azul.

El color negro no tiene ninguna luminosidad, así que todos los bits valen Ø (off). Por tanto el código para el negro en binario es ØØØ, es decir, cero.

Los colores puros, verde, rojo y azul tienen solamente un bit que vale 1 (on) de un total de tres posiciones. Sus códigos son 1ØØ, Ø1Ø y ØØ1 en binario, es decir, cuatro, dos y uno.

Los demás colores son mezclas de éstos, así que sus códigos en binario tienen dos ó más bits 1.